Evaluation quantitative de la sécurité informatique : approche par les vulnérabilités

Géraldine Vache Marconato

THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Institut National des Sciences Appliquées de Toulouse*
Discipline ou spécialité : *Système Informatiques*

Présentée et soutenue par *Géraldine Vache-Marconato*
Le *8 décembre 2009*

Titre : *Evaluation quantitative de la sécurité informatique : approche par les vulnérabilités*

JURY
Jean-Claude Laprie : Directeur de thèse
Marc Dacier : Rapporteur
Brian Randell : Rapporteur
Michel Cukier : Examinateur
Vincent Nicomette : Examinateur

Ecole doctorale : *Ecole Doctorale Systèmes*
Unité de recherche : *LAAS-CNRS*
Directeur(s) de Thèse : *Jean-Claude Laprie*
Rapporteurs : *Marc Dacier et Brian Randell*

Remerciements

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS). Je remercie Malik Ghallab et Raja Chatila, qui ont assuré la direction du LAAS-CNRS depuis mon entrée.

Je remercie également Jean Arlat et Karama Kanoun, responsables successifs du groupe de recherche Tolérance aux fautes et sûreté de fonctionnement informatique, pour m'avoir permis de réaliser mes travaux au sein du groupe.

Je tiens à exprimer ma profonde reconnaissance et mon grand respect à Jean-Claude Laprie, Directeur de Recherche de Classe Exceptionnelle au CNRS, qui a accepté d'encadrer mes travaux. Son immense culture du domaine, ses conseils pertinents et ses critiques constructives m'ont permis de toujours chercher à me remettre en question et à rester rigoureuse, permettant à cette thèse d'être menée à terme.

J'exprime ma gratitude et remercie chaleureusement Mohamed Kaâniche, Directeur de Recherche au LAAS-CNRS pour son soutien et ses conseils avisés. Malgré son emploi du temps plus que rempli, sa porte m'a toujours été ouverte.

Je tiens à exprimer ma profonde reconnaissance et ma sincère amitié à Vincent Nicomette, Maître de Conférences à l'INSA de Toulouse et psychologue pour doctorante pleine de doute à ses moments perdus. Sa patience, sa disponibilité, son amitié et son intérêt pour mes "boules et flèches" lors des innombrables heures de discussion m'ont permis d'avancer et de ne pas me décourager dans les moments difficiles.

J'exprime ma gratitude à :

Jean-Claude Laprie, Directeur de Recherche au LAAS-CNRS,

Marc Dacier, Directeur du laboratoire Symantec Europe à Sophia Antipolis,

Brian Randell, Professeur Emerite à l'Université de Newcastle upon Tyne, Royaume-Uni,

Michel Cukier, Professeur à l'Université du Maryland, Etats-Unis,

Vincent Nicomette, Maître de conférences à l'INSA de Toulouse

pour l'honneur qu'ils me font en constituant à mon jury.

Je remercie tout particulièrement Marc Dacier et Brian Randell qui ont accepté la charge d'être rapporteurs et qui ont consacré du temps à discuter avec moi du contenu de ce mémoire. Je tiens spécialement à exprimer ma gratitude à Brian Randell pour qui la langue dans laquelle est écrit ce mémoire a représenté un surcroît de travail.

Je tiens à remercier tous ceux qui ont partagé au sein du laboratoire ces trois années de thèse, en commençant naturellement par les doctorants du groupe TSF, avec une mention particulière pour ceux qui m'ont supportée plus que les autres : Ossama, les deux Eric, Youssef, Bob, Ana, Thomas, Rim, Fernand, Anthony et tous ceux que je m'excuse déjà d'avoir oublié dans cette liste. Je remercie également toutes les personnes qui permettent aux membres du laboratoire d'avoir un cadre de travail agréable, de la documentation au magasin en passant par la reprographie, sans oublier Gina Briand, qui a su me sortir de tous les mystérieux problèmes administratifs et logistiques que j'ai rencontrés au cours de ces trois ans.

Je remercie également Vincent Nicomette, Ernesto Exposito, Christophe Chassot et l'ensemble du corps enseignant du département DGEI de l'INSA de Toulouse, qui m'ont fait confiance et permis de faire de l'enseignement durant la préparation de cette thèse. Ces heures passées à préparer ou animer des cours m'ont permis de trouver un équilibre dans mon quotidien de doctorante.

J'adresse un grand merci à Mathias et "Momo", ATER à l'INSA de Toulouse, pour leur bonne humeur et le soutien qu'ils m'ont apporté durant ces trois ans. Merci également à Slim Abdellatif, Christophe Zanon, et tous ceux dont le sourire a été si communicatif.

Enfin, cette thèse n'aurait sans aucun doute jamais eu lieu sans le soutien sans faille de ma famille (surtout à "Sœurette", dont le nombre de mails par jour a pu tripler dans les moments de doute ou d'euphorie) et des copains : les "pichos", les (désormais "vieux") enfoiros, les "graines de sel" et tous les autres avec une mention spéciale pour Sébastien qui commence tout juste sa propre thèse. Ils ont toujours accepté et excusé mes trop nombreuses absences ou bien la présence quasi perpétuelle à mes cotés de mon ordinateur et de ma pile de documents. Enfin, il m'est impossible de ne pas citer Benjamin, qui sait déjà tout ce que je n'oserai pas écrire sur ces pages. Je les remercie tous du fond du cœur. Cette thèse est la leur.

Table des matières

Introduction

En une quinzaine d'années, l'importance de l'ordinateur dans le quotidien a connu un essor considérable. L'ordinateur personnel muni d'une connexion Internet est devenu quasi indispensable dans n'importe quel foyer. On comptait 1,45 milliard d'internautes en 2008 (source : *Journal du Net*[1]). En 2006, 55% de la population française possédait un ordinateur personnel (Source : *Journal du Net*[2]), contre seulement 15% en 1996 (source : *INSEE*[3]). Parmi eux, les 2/3 possèdent une connexion à Internet. Bien sûr, ces chiffres ne font pas référence à la progression fulgurante depuis l'arrivée sur le marché des téléphones portables troisième génération. En plus d'avoir à sa disposition une source d'information immense, un internaute dispose d'un grand panel de services : de la consultation de son compte en banque à la communication par mail ou messagerie instantanée ou encore le commerce en ligne, Internet est devenu incontournable pour ceux qui l'utilisent.

Il en est bien sûr de même dans le milieu professionnel où le domaine de production d'applications pour les systèmes d'information s'est développé de manière fulgurante. Pour les entreprises, offrir les services via Internet est devenu essentiel : chaque banque possède son site de gestion en ligne - 46% des internautes français ont eu recours à

[1] http://www.journaldunet.com/chiffres-cles.shtml

[2] http://www.journaldunet.com/cc/02_equipement/equip_pc_fr.shtml

[3] http://www.insee.fr/fr/ffc/docs_ffc/IP1011.pdf

ce type de service en août 2008 (source : *comScore*[4]). Les sites de vente en ligne deviennent indénombrables et il est ainsi devenu possible pour une boutique de faire affaire avec un client à l'autre bout du monde.

Qu'il s'agisse donc du système d'information professionnel ou de l'ordinateur personnel que chacun a chez soi, cette démocratisation du système d'information et des services qu'il offre, via Internet ou non, a entraîné entre autres deux conséquences importantes.

La première est l'augmentation du nombre de vulnérabilités présentes sur ces systèmes d'information. En multipliant les applications disponibles, le nombre de vulnérabilités a, lui aussi, augmenté de façon très significative : le nombre de vulnérabilités recensées en 1997 était de 311 alors qu'il a atteint 7236 en 2007 (source : *CERT Coordination Center*[5]). Ces vulnérabilités de plus en plus nombreuses sur les systèmes d'information ont eu pour conséquence directe de rendre ces systèmes plus exposés aux activités des pirates.

De plus, l'accès à une machine distante permis par une connexion réseau, comme par exemple le réseau Internet, facilite le travail de cette population malveillante et permet de nouveaux types d'attaque vis-à-vis de la sécurité des systèmes. La considérable augmentation de l'utilisation de l'Internet a donc multiplié également les possibilités d'attaque, ainsi que les motivations des personnes malveillantes : vol d'information, d'argent, d'identité, dénis de service ou encore défiguration de site Web... La liste n'est pas exhaustive.

Cette augmentation de nombre de vulnérabilités est liée à la complexité croissante des applications, mais également aux exigences de production. De plus, il n'est pas encore possible, à l'heure actuelle, de produire un logiciel complexe sans faute à partir d'un modèle formel. Il est donc nécessaire d'utiliser des techniques permettant de prévoir et d'éliminer les fautes durant la phase de conception et de développement du système, avant que celui-ci ne soit diffusé sur le marché. Malheureusement, ces méthodes ne s'avèrent pas suffisantes et ne sont pas toujours respectées pour répondre à des impératifs de coût et de temps de production. De plus, les méthodes de génie logiciel et de conception n'ont pas évolué au même rythme que la complexité des systèmes. Face à ces problèmes, il est donc impératif d'agir durant la phase de vie opérationnelle du système.

Le système en opération ainsi vulnérable est déployé dans un environnement qui lui est, en partie, propre et qui, de plus, évolue. Cet environnement va avoir une influence certaine sur le système et donc sur sa sécurité. Le point de départ de nos travaux est de

[4] http://www.comscore.com/Press_Events/Press_Releases/2008/10/Online_Banking_in_France/%28language%29/eng-US

[5] http://www.cert.org/stats

nous intéresser à l'environnement du système vis-à-vis des vulnérabilités et d'étudier comment ce dernier évolue. Notre but est de prendre en considération à la fois le système et son environnement pour mesurer certains aspects de la sécurité du système, à partir de deux points de vue possible : 1) l'effort à fournir pour atteindre le niveau de sécurité voulu ; 2) les risques encourus en considérant un niveau de sécurité donné. L'approche que nous avons mis au point a pour but de produire des mesures quantitatives - telles que la probabilité de compromission du système avant une période donnée - mais doit également être capable de s'adapter à ces deux points de vue.

Nos travaux s'inscrivent dans le cadre du réseau d'excellence européen ReSIST (*Resilience for Survivability in Information System Technologies* en anglais). Ce réseau d'excellence, débuté en 2006, a permis une collaboration entre plusieurs laboratoires et universités académiques pour mener une réflexion sur les problématiques liées à la résilience. Ce réseau d'excellence avait quatre objectifs : 1) permettre à plusieurs équipes de recherche multidisciplinaires de réfléchir sur les sujets fondamentaux concernant l'évolutivité des systèmes résilients ; 2) identifier des directions de recherche concernant la confiance placée dans ces systèmes ; 3) la production de résultats significatifs, briques de base pour assurer la résilience des systèmes ; 4) la promotion et la propagation des travaux et concepts du domaine de la résilience dans la culture académique et industrielle.

Notre mémoire présente nos travaux selon le plan suivant. Dans un premier chapitre, nous rappelons les grands concepts liés aux domaines de la sûreté de fonctionnement et de la sécurité informatique afin de préciser la terminologie et les concepts utilisés dans ce mémoire. Ce chapitre présente également un panorama des travaux existants dans le domaine de l'évaluation de la sécurité afin de placer notre approche dans le paysage des approches quantitatives pour la mesure de la sécurité.

Le second chapitre de notre mémoire présente plus précisément le cadre de nos travaux et la démarche que nous adoptons pour la production des mesures. Nous y spécifions les concepts spécifiques que nous avons utilisés et présentons les facteurs environnementaux que nous avons identifiés : 1) le cycle de vie de la vulnérabilité; 2) le comportement de la population des attaquants et 3) le comportement de l'administrateur – ainsi que leur évolution comportementale dans le temps. Nous en déduisons les scénarios d'exploitation de vulnérabilité qu'il nous est indispensable de distinguer afin d'élaborer le processus de mesure.

Ce processus se base sur une modélisation du système et de son environnement. Elle est présentée dans le troisième chapitre. Nous détaillons dans ce chapitre les modèles que nous avons produits. Dans un premier temps, nous avons défini un modèle produisant des mesures telle que la probabilité de compromission du système par l'exploitation d'une vulnérabilité. Nous avons quantifié les événements d'après les études menées et diffusées dans la littérature. Nous décrivons par la suite la validation de ce modèle par l'étude d'une vulnérabilité.

Le quatrième chapitre de ce mémoire décrit les dernières étapes de notre démarche. En effet, la quantification des événements modélisés dans notre approche doit être affinée afin de rendre les mesures résultantes plus fines et plus réalistes. Nous présentons donc dans ce chapitre notre étude pour la quantification des événements caractérisant certains aspects de l'environnement du système tels que le phénomène de publication de la vulnérabilité ou encore la mise au point du kit d'exploitation.

Dans le cinquième et dernier chapitre, nous présentons une extension de notre modélisation pour pouvoir prendre en considération plusieurs vulnérabilités en parallèle. Nous étudions les dépendances qui peuvent exister dans ce contexte. La seconde partie de

notre chapitre présente les simulations menées pour valider à la fois cette étude et l'extension de notre modélisation pour plusieurs vulnérabilités.

Enfin, dans une conclusion, nous présentons un bilan de nos travaux. Nous essayons d'avoir un recul critique sur notre approche et détaillons les améliorations possibles ainsi que les perspectives que nous envisageons à court et moyen terme.

Chapitre 1 : Historique des mesures dans le domaine de la sécurité

Dans ce premier chapitre, nous présentons d'abord le domaine de la sûreté de fonctionnement puis plus précisément celui de la sécurité informatique, qui réunit trois des principaux attributs de la sûreté de fonctionnement : confidentialité, intégrité et disponibilité. Ensuite, nous étudions la problématique de la mesure et de l'évaluation dans le cadre de la sécurité et faisons un panorama de la taxonomie rencontrée dans le contexte d'évaluation de la sécurité. Enfin, nous présentons les principales approches pour l'évaluation de la sécurité publiées à ce jour. Notre chapitre se termine par une première introduction à notre approche.

1 La sécurité-immunité

1.1 La sûreté de fonctionnement : concepts et définitions

1.1.1 Définition de base et attributs

Dans [LAP 95, LAP 04, AVI 04], les auteurs donnent la définition de la sûreté de fonctionnement : la **sûreté de fonctionnement** d'un système informatique est l'aptitude à délivrer un service de confiance justifiée.

Le service délivré par un système est son comportement tel qu'il est perçu par son ou ses utilisateurs, l'utilisateur étant un autre système, humain ou non.

Selon la ou les applications auxquelles le système est destiné, l'accent peut être mis sur différentes facettes de la sûreté de fonctionnement, ce qui revient à dire que la sûreté de fonctionnement peut être vue selon des propriétés différentes mais complémentaires qui permettent de définir ses attributs :

- Le fait d'être prêt à l'utilisation conduit à la **disponibilité** (*availability* en anglais) ;
- La continuité du service conduit à la **fiabilité** (*reliability* en anglais) ;
- L'absence de conséquences catastrophiques conduit à la **sécurité-innocuité** (*safety* en anglais) ;
- L'absence de divulgations non autorisées de l'information conduit à la **confidentialité** (*confidentiality* en anglais) ;
- L'absence d'altérations inappropriées de l'information conduit à l'**intégrité** (*integrity* en anglais) ;
- L'aptitude aux réparations et aux évolutions conduit à la **maintenabilité** (*maintenability* en anglais).

L'association de la confidentialité, l'intégrité et la disponibilité vis-à-vis des actions autorisées conduit à la sécurité-immunité, que nous appellerons simplement sécurité dans la suite de ce mémoire. Ces attributs sont résumés sur la figure 1.1.

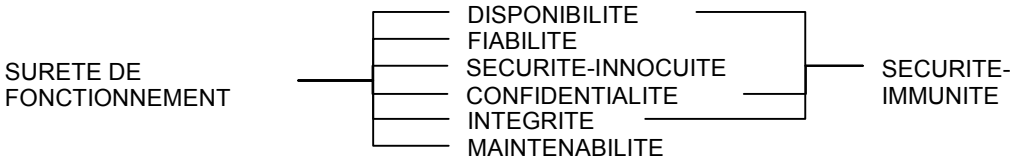


Figure 1.1 : Les attributs de la sûreté de fonctionnement

Les travaux décrits dans ce mémoire traitent plus particulièrement de la sécurité-immunité.

1.1.2 Notions de faute, erreur et défaillance

Ces trois notions sont appelées les entraves à la sûreté de fonctionnement. Une défaillance du service survient lorsque le service délivré par le système dévie du service correct. Rares sont les systèmes qui ne défaillent pas, d'où cette définition alternative de la sûreté de fonctionnement [LAP 04] : la sûreté de fonctionnement est l'aptitude à éviter des défaillances du service plus fréquentes ou plus graves qu'acceptables.

La partie de l'état du système pouvant entraîner une défaillance est une erreur. La cause adjugée ou supposée d'une erreur est une faute. Dans ce paragraphe, nous présentons ces trois notions et les liens entre elles.

Un système ne défaille pas toujours de la même façon, ce qui conduit à la notion de mode de défaillance, qu'il est possible de caractériser selon quatre points de vue :

- Le **domaine** : une défaillance peut être une défaillance en valeur (la valeur du service délivré ne permet plus l'accomplissement de la fonction du système) ou une défaillance temporelle (les conditions temporelles de délivrance du service ne permettent plus l'accomplissement de la fonction du système) ;
- La **détectabilité** : une défaillance peut être signalée (le service délivré est détecté comme incorrect, et signalé comme tel) ou non signalée (la délivrance d'un service incorrect n'est pas détectée) ;
- La **cohérence** : une défaillance peut être cohérente (le service incorrect est perçu identiquement par tous les utilisateurs) ou byzantine (certains ou tous les utilisateurs perçoivent différemment le service incorrect) ;
- Les **conséquences** : de la défaillance bénigne (les conséquences sont du même ordre de grandeur que le bénéfice procuré par le service délivré en l'absence de défaillance) à la défaillance catastrophique (les conséquences sont incommensurablement différentes du bénéfice procuré par le service délivré en l'absence de défaillance).

Quant aux fautes, elles peuvent être considérées selon huit axes de classification élémentaires :

- La **phase de création ou d'occurrence** distingue les fautes de développement, qui sont créées durant le développement du système ou la maintenance en vie opérationnelle, des

fautes opérationnelles, qui surviennent durant les périodes de délivrance du service au cours de la vie opérationnelle ;

- Les **frontières du système** distinguent les fautes internes qui sont localisées à l'intérieur des frontières du système, des fautes externes, localisées à l'extérieur des frontières du système et pouvant propager des erreurs à l'intérieur du système ;
- La **cause phénoménologique** distingue les fautes naturelles dues à des phénomènes naturels et sans intervention humaine directe des fautes dues à l'homme ;
- La **dimension** distingue les fautes matérielles des fautes logicielles ;
- L'**objectif** distingue les fautes malveillantes, c'est-à-dire introduites par un humain avec l'objectif malveillant de causer des dommages au système, des fautes non malveillantes, c'est-à-dire introduites sans objectif malveillant ;
- L'**intention** distingue les fautes délibérées, résultant d'une décision, des fautes non délibérées ne résultant pas d'une décision consciente ;
- La **capacité** distingue les fautes d'incompétence, qui résultent d'un manque de compétence professionnelle ou de l'inadéquation de l'organisation du développement du système, des fautes accidentelles, créées par inadvertance ;
- La **persistance** distingue les fautes permanentes, dont la présence est continue, des fautes transitoires dont la présence est temporairement bornée.

Ces classes de fautes élémentaires permettent de caractériser une faute et de considérer des classes de fautes combinées, regroupées en trois familles : les fautes de développement, les fautes physiques et les fautes d'interaction.

Une faute est active lorsqu'elle produit une erreur. Une faute active est soit une faute interne qui était préalablement dormante et qui a été activée par le processus de traitement, soit une faute externe. Une faute interne peut alterner entre les états dormant et actif.

Une erreur peut être latente tant qu'elle n'a pas été reconnue en tant qu'erreur, ou détectée par un algorithme ou mécanisme de détection car elle produit une défaillance. Une erreur latente peut être corrigée avant d'être détectée.

Une défaillance survient lorsqu'une erreur affecte le service délivré par le système, donc lorsqu'elle traverse l'interface système-utilisateur. La conséquence de la défaillance d'un composant est une faute interne pour le système et une faute externe pour les composants avec lesquels il interagit, comme résumé sur la figure 1.2.

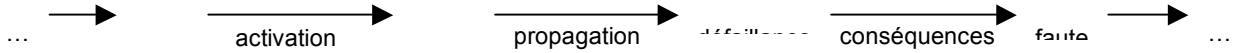


Figure 1.2 : Les entraves de la sûreté de fonctionnement

1.1.3 Les moyens pour la sûreté de fonctionnement

Les moyens de la sûreté de fonctionnement sont les méthodes et techniques permettant de fournir au système l'aptitude à délivrer un service conforme à l'accomplissement de sa fonction, et de donner confiance dans cette aptitude. Cet ensemble de méthodes et de techniques peut être classé en quatre catégories :

- La **prévention des fautes** : comment empêcher l'occurrence ou l'introduction de fautes ;
- La **tolérance aux fautes** : comment fournir un service à même de remplir la fonction du système en dépit des fautes ;
- L'**élimination des fautes** : comment réduire la présence (le nombre, la sévérité) des fautes ;
- La **prévision des fautes** : comment estimer la présence, la création et les conséquences des fautes.

Seule l'utilisation combinée de ces moyens peut conduire à un système qui soit sûr de fonctionnement. Ainsi, l'association entre élimination de fautes et prévision des fautes peut être considérée comme la validation de la sûreté de fonctionnement, c'est-à-dire comment avoir confiance dans l'aptitude du système à délivrer un service conforme à l'accomplissement de sa fonction. De même, l'élimination des fautes est souvent étroitement associée à la prévention des fautes, l'ensemble constituant l'évitement des fautes, c'est-à-dire tendre vers un système sans faute.

1.1.4 Le concept de résilience

Le concept de résilience, apparu il y a de nombreuses années déjà dans le domaine de la sûreté de fonctionnement, est de plus en plus utilisé. Dans [LAP 08], l'auteur fait une synthèse des domaines et des définitions rencontrées et propose la définition suivante de la résilience : la résilience est la persistance de la sûreté de fonctionnement en présence de changements. Ces changements sont caractérisés selon trois axes de classification :

- La **nature** du changement : il peut être fonctionnel, environnemental ou technologique ;
- La **perspective** du changement : il peut être prévu, prévisible ou non prévu ;
- L'**échéance** du changement : il peut être à court terme (de l'ordre de la seconde à celui de l'heure), à moyen terme (de l'ordre de l'heure à celui de plusieurs mois) ou à long terme (de l'ordre du mois à celui de plusieurs années).

1.2 La sécurité : le domaine des malveillances

La sécurité informatique est un terme large qui réunit les moyens humains, techniques, organisationnels et juridiques qui tentent de garantir certaines propriétés d'un système d'information. Les moyens mis en place pour assurer la sécurité d'un système d'information peuvent être de plusieurs natures. Assurer un niveau de sécurité pour le système est donc important au niveau logiciel (gestion et protection des données, des applications et des communications réseaux). C'est ce premier aspect de la sécurité auquel nous nous intéresserons plus particulièrement. Mais, il ne faut pas négliger un aspect important pour la sécurité du système qui est la protection physique du matériel (contre les vols, les dégradations physiques volontaires ou involontaires comme les incendies ou les inondations, etc.). La sécurité d'un système d'information est donc une notion large, qui peut néanmoins se définir en trois attributs principaux, rappelés dans le paragraphe suivant.

1.2.1 Les attributs de la sécurité

Comme nous l'avons évoqué précédemment, la sécurité est l'association de trois des attributs principaux de la sûreté de fonctionnement qui sont la confidentialité, l'intégrité et la disponibilité, que nous avons définis dans le paragraphe 1.1.1.

La confidentialité définit l'absence de divulgation non autorisée de l'information. Une attaque contre la confidentialité par une personne malveillante consiste à tenter de récupérer des informations pour lesquelles elle ne possède pas d'autorisation, soit en tentant d'y accéder sur le système, soit en écoutant les communications réseaux, soit de toute autre façon possible..

L'intégrité définit l'absence d'altération inappropriée de l'information. Une attaque contre l'intégrité vise à introduire de fausses informations, ou à modifier ou détruire l'information existante. Tout comme pour la confidentialité, l'attaquant peut, par exemple, chercher à atteindre l'information directement sur le système ou à l'intercepter durant une communication.

La disponibilité définit le fait que le système soit 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.

En plus de ces trois attributs, la sécurité compte des attributs dits secondaires, que nous détaillons ici :

- La **non répudiation** (*nonrepudiability* en anglais) : regroupe la disponibilité et l'intégrité de l'identité de l'émetteur d'un message (non réfutation de l'origine) ou du destinataire d'un message (non réfutation de la destination) ;
- L'**authenticité** (*authenticity* en anglais) : regroupe l'intégrité du contenu et de l'origine d'un message, et éventuellement d'autres informations, comme l'instant d'émission ;
- La **responsabilité** (*accountability* en anglais) : regroupe la disponibilité et l'intégrité de l'identité de la personne qui a effectué une opération.

1.2.2 Terminologie dans le domaine de la sécurité

Le domaine de la sécurité possède également ses propres termes, qui ne sont pas utilisés dans le domaine de la sûreté de fonctionnement. Ce sont ces termes que nous évoquons dans ce paragraphe.

Le premier terme spécifique à la sécurité est la **vulnérabilité**. La définition donnée dans [AVI 04] rattache la vulnérabilité au concept de faute : une vulnérabilité est une faute interne qui permet à une faute externe d'endommager le système. Cette définition peut être complétée par celle élaborée dans le cadre du projet MAFTIA[6] [MAF 01, MAF 03] : une vulnérabilité est une faute accidentelle, ou intentionnelle, malveillante ou non, dans les spécifications, la conception ou la configuration du système, ou dans la manière dont il est utilisé, qui peut être exploitée pour créer une intrusion. Cette définition rejoint la précédente et définit la vulnérabilité comme une entrave à la sécurité. On retrouve dans la norme ISO:17799 – devenue ISO:27002 – la définition suivante : une vulnérabilité est une faiblesse d'un bien (quelque chose ayant de la valeur pour l'organisation, ses opérations et leur continuité) ou groupe de biens qui peut être exploitée par un attaquant [ISO 00]. Dans cette définition intervient la notion d'attaque, également présente dans la définition de [WHI 03] qui définit une vulnérabilité comme étant la propriété d'un objet du système de permettre une attaque. Dans [NRC 02], une vulnérabilité est définie comme une erreur ou une faiblesse dans la conception, l'implémentation ou le fonctionnement du système. Ces définitions sont en accord avec l'une des premières définitions connues dans [AND 80], dans laquelle l'auteur définit une vulnérabilité comme une faille connue ou suspectée, issue de la conception ou l'opération, du matériel ou du logiciel qui expose le système à une intrusion ou qui permet l'exposition accidentelle d'informations. Enfin, l'entreprise Microsoft possède une définition plus énumérative en mettant en avant les différentes possibilités ainsi offertes à un attaquant : une vulnérabilité est une faille dans un produit qui offre la possibilité à un attaquant d'usurper des privilèges d'un utilisateur, effectuer des opérations, compromettre des données, accéder à des données confidentielles. Malgré cela, cette liste non exhaustive de définitions semblent cohérente et définit le même concept global. Dans [TRA 05], l'auteur propose une définition différente des précédentes : il définit une vulnérabilité comme un ensemble de conditions ou de failles susceptibles de favoriser une agression (attaque sur une cible). Cette définition est différente dans le sens où la vulnérabilité est un ensemble de faiblesses. Ce qui est désigné ici comme une faille est ce qui semble avoir été défini plus haut sous le terme de vulnérabilité.

Dans certaines des définitions que nous avons citées apparaît la notion d'attaque ou celle d'attaquant. Dans [MAF 01, MAF 03], une **attaque** est définie comme faute d'interaction malveillante visant à violer une ou plusieurs propriétés de sécurité. C'est une faute externe créée avec l'intention de nuire, y compris les attaques lancées par des outils automatiques : vers, virus, zombies, etc. La notion d'attaque ne doit pas être confondue avec la notion d'intrusion. Dans [MAF 01, MAF 03], il est précisé qu'un système peut être

[6] Le projet MAFTIA a porté sur le développement de techniques de tolérance aux fautes accidentelles et aux malveillances pour des applications réparties à grande échelle sur Internet. En particulier, une terminologie des malveillances a été introduite dans le cadre de ce projet.

attaqué sans que cette attaque soit couronnée de succès. Dans ce cas de figure, il n'y a pas eu d'intrusion. Une **intrusion** est donc définie comme une faute malveillante interne d'origine externe, résultant d'une attaque qui a réussi à exploiter une vulnérabilité. Elle est susceptible de produire des erreurs pouvant provoquer une défaillance vis-à-vis de la sécurité, c'est-à-dire une violation de la politique de sécurité du système.

Dans la suite, nous parlerons donc d'attaque comme une tentative délibérée et malveillante exploitant une vulnérabilité. Le terme d'intrusion sera employé dans le cas où l'attaque est menée avec succès et où l'attaquant a réussi à introduire et/ou compromettre le système.

1.2.3 Les moyens pour la sécurité

Les moyens décrits dans le paragraphe précédent pour la sûreté de fonctionnement sont des moyens pouvant également satisfaire les attributs de la sécurité. Dans ce paragraphe, nous présentons les quatre moyens pour la sûreté de fonctionnement - la prévention des fautes, la tolérance aux fautes, l'élimination des fautes, et la prévision des fautes - appliqués au domaine de la sécurité.

1.2.4 La prévention des fautes

La prévention des fautes a pour objectif d'empêcher l'occurrence ou l'introduction de fautes dans le système. Ce travail a lieu durant la première phase du cycle de vie du système : la phase de conception et d'implémentation. Il englobe les bonnes pratiques d'ingénierie et l'utilisation de politiques de sécurité, présentées dans la suite.

Parmi les bonnes pratiques d'ingénierie, la spécification formelle occupe une place importante. Elle repose sur des bases mathématiques pour décrire ce que doit faire le système – et non pas comment il doit le faire. Une validation est réalisée pour s'assurer que la description coïncide effectivement avec les attentes. Cette approche formelle permet de lever les ambiguïtés qui peuvent persister dans une spécification en langage naturel. La description peut être réalisée avec la notation Z ou la méthode B, par exemple. Elle sera utilisée à son tour lors de l'étape de développement. Idéalement, le développement du système doit être, dans la mesure du possible, une simple traduction en langage de programmation de la description du système. Concernant le développement, des normes ont été mises en place pour des secteurs d'activités sensibles. Elles imposent des règles dans l'utilisation des langages de programmation. Le but est de prévenir les fautes fréquentes. Par exemple, dans la norme MIRA-C [MIR 04] pour l'automobile, l'arithmétique sur les pointeurs est proscrite. Cette pratique évite l'introduction de fautes par ces techniques avancées de programmation, ainsi que leur utilisation malveillante. Ces techniques ont été initialement appliquées pour faire face aux fautes accidentelles, par exemple dans le logiciel. Elles restent généralement applicables dans le cas des malveillances. Dans ce domaine, un effort important a également été consacré au développement de politiques de sécurité.

L'ensemble des propriétés de sécurité que l'on désire assurer dans un système ainsi que la façon dont ces propriétés vont être assurées sont définies dans la politique de sécurité du système. "La politique de sécurité d'un système est l'ensemble des lois, règles et pratiques qui régissent la façon dont l'information sensible et les autres ressources sont gérées, protégées et distribuées à l'intérieur d'un système spécifique" [ITS 91]. Elle doit identifier les objectifs de sécurité et les menaces auxquelles ceux-ci devront faire face. Cette notion de politique de sécurité peut être raffinée en trois branches distinctes : les

politiques de sécurité physique, administrative et logique. La première s'occupe de tout ce qui touche à la situation physique du système à protéger. En particulier, y sont définies les mesures contre le vol par effraction, les incendies, les catastrophes naturelles, etc. Les politiques administratives traitent de tout ce qui touche à la sécurité d'un point de vue organisationnel dans l'entreprise comme la sélection du personnel responsable de la sécurité du parc informatique de l'entreprise. La politique de sécurité logique s'intéresse au contenu du système informatique. Elle doit spécifier tous les contrôles d'accès logiques, et doit spécifier qui a accès à quoi et en quelles circonstances. Elle peut se décomposer en plusieurs phases. Chaque individu qui utilise un système sécurisé doit s'identifier et doit pouvoir prouver qu'il est bien la personne qu'il prétend être. Ces deux phases sont définies respectivement dans la politique d'identification et la politique d'authentification. Un utilisateur ayant fourni un mot de passe valide est authentifié et peut alors accéder au système. La politique d'autorisation décrit les opérations que cet utilisateur peut alors réaliser dans le système.

Dans [CUP 06] est fournie une description récente et plus détaillée des modèles et politiques de sécurité.

1.2.4.1 La tolérance aux fautes

La tolérance aux fautes correspond à un ensemble de moyens destiné à assurer qu'un système remplit sa fonction en dépit des fautes. Elle est obtenue grâce à la mise en œuvre de techniques de traitement d'erreurs et de fautes. La première étape pour la tolérance aux fautes est la détection d'erreur, qui permet d'identifier un état erroné. La seconde étape est le rétablissement du système qui permet de le ramener dans un état sûr. Du point de vue des fautes malveillantes, un système tolérant aux intrusions est un système capable de s'auto-diagnostiquer, de se réparer et de se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes même pendant les attaques [DES 91]. Dans ce paragraphe, nous présentons les principales approches pour la détection d'intrusions puis pour la tolérance aux intrusions.

Les méthodes de détection des intrusions visent à détecter des atteintes à la politique de sécurité d'un système. Dans ces travaux, la surveillance des activités du système est suggérée pour identifier les activités malveillantes. Etant donne la quantité importante de données à traiter, ces méthodes sont actuellement automatisées par l'emploi de systèmes de détection d'intrusions (*Intrusion Detection System*, en anglais). Ces systèmes de détection d'intrusions peuvent être passifs (c'est-à-dire se contenter de notifier la présence d'une intrusion) ou bien actif (c'est-à-dire réagir à la détection de l'intrusion pour la stopper) [GAD 08]. Par ailleurs, il existe deux approches pour la détection des intrusions : l'approche comportementale et l'approche par scénario. L'approche comportementale répond à la question : "le comportement actuel de l'utilisateur est-il cohérent avec son comportement passé ?". Elle repose sur une base de connaissances des comportements qui peut être enrichie par un apprentissage permanent. L'approche par scénario utilise une base de signatures de scénarios anormaux. Elle détecte une atteinte à la politique de sécurité lorsqu'une séquence d'informations – liée à un comportement – possède une signature référencée dans la base de signatures des scénarios anormaux. Confronter les comportements observés à la base de signatures de comportements anormaux est difficile. De plus, la base doit être constamment mise à jour afin de pouvoir détecter les scénarios décrivant les nouvelles attaques connues. Cette approche a été implémentée par exemple dans l'outil *snort* [SNO 09]. Cependant, les outils de détection d'intrusions génèrent encore de nombreuses alarmes sans intrusion réelle, ce sont des fausses alarmes ou faux positifs ; par ailleurs, il ne parviennent pas à détecter certaines atteintes à la politique de sécurité, appelées faux négatifs. Certains travaux proposent donc

l'utilisation conjointe de plusieurs *IDS* [MEM 01, CUP 02]. Un état de l'art récent sur les techniques de détection d'intrusion est présenté dans [DAC 06].

Nous avons vu qu'un système tolérant aux intrusions devait pouvoir s'auto-diagnostiquer, se réparer et se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes pendant les attaques. Nous avons décrit dans le paragraphe précédent comment le système pouvait détecter les intrusions. Mais quelles sont les techniques existantes pour tolérer ces intrusions ? Parmi les techniques classiques dans le domaine de la sécurité des systèmes, on trouve le chiffrement, la réplication, et le brouillage des données. Une autre technique originale développée au LAAS utilise les concepts de fragmentation, redondance et dissémination [DES 91]. C'est une approche globale qui assure la tolérance aux fautes accidentelles aussi bien qu'intentionnelles pour le stockage, le traitement et la transmission de données confidentielles. Le principe est le suivant : 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. Le système global n'assurera donc pas un service correct que si la majorité des sous-systèmes est défaillante [SAI 05, SAI 09].

Une des premières architectures tolérantes aux intrusions a été développée dans le cadre du projet DELTA-4 dans les années 1980, pour des serveurs de stockage de données, d'authentification et d'autorisation. Plus récemment, les principes de tolérance aux intrusions ont été employés dans d'autres projets, comme [VER 06]. Les travaux réalisés dans le projet MAFTIA ont visé à développer des politiques d'autorisation plus efficaces en se basant sur des techniques de fragmentation, redondance et dissémination et les cartes à puces Java. Les politiques proposées dans ce projet permettent de fournir des systèmes tolérants aux fautes accidentelles et aux intrusions.

1.2.4.2 L'élimination des fautes

Malgré la prévention des fautes, certaines fautes introduites dans le système durant la phase de conception et d'implémentation peuvent avoir persisté. Une fois le système en phase opérationnelle, ces fautes pourront être exploitées à des fins malveillantes. L'élimination des fautes est le moyen de la sûreté de fonctionnement qui vise à réduire le nombre de fautes en utilisant des techniques de vérification et de test.

La vérification formelle est une technique permettant de vérifier que les propriétés attendues sont satisfaites conformément aux hypothèses. Elle se base sur une description formelle du système et une liste des propriétés à vérifier. Une approche courante consiste à utiliser des outils de *model-checking* comme SMV [SMV 01]. De plus, certaines méthodes permettent l'analyse du code source afin de détecter des fautes particulières, telles que les dépassements de tampon [ALL 07, HAU 03, LAR 01]. Cependant, la vérification formelle doit s'accompagner d'un diagnostic de localisation de la faute pour permettre la mise au point d'un correctif.

Le test est une technique largement utilisée : il s'agit d'appliquer au système des valeurs d'entrée correspondant à des scénarios d'utilisation normaux ou anormaux et d'en déduire le bon respect ou non des propriétés que doit assurer le système en fonction des

sorties observées. Différents types de test existent en fonction des critères considérés pour la sélection et la génération des entrées. En particulier, le test de robustesse, basé sur l'injection de fautes, permet de tester la réaction du système dans des conditions extrêmes et en présence d'entrées valides ou invalides.

Dans le domaine de la sécurité, outre les techniques classiques de test vis-à-vis des fautes de conception, des techniques spécifiques ont été développées pour l'analyse des vulnérabilités, comme par exemple les tests de pénétration développés par les experts en sécurité (appelés *red team* ou *tiger team*) qui tentent de violer les objectifs de sécurité en contournant les mécanismes de protection. Enfin, une technique d'élimination des fautes dans le cadre de la sécurité est la vérification de la configuration du système et de la présence de vulnérabilités par une analyse du système de fichiers à l'aide d'une base de données de vulnérabilités connues. Des outils de recherche de telles vulnérabilités ont été développés tels que *Nessus* [LAU 02], *Cops* [FAR 90] ou encore *Esope* [ORT 99].

1.2.4.3 La prévision des fautes

La prévision des fautes a pour objet l'identification des fautes, erreurs et modes de défaillances potentiels du système. Cette analyse conduit à un processus d'évaluation de l'impact de ces fautes. Dans le cadre de la sécurité, de nombreux travaux ont été menés en ce sens. C'est ce que nous présentons dans le paragraphe suivant.

2 Evaluation de la sécurité

Dans le domaine de la sécurité, différents types d'évaluation ont été développés : 1) les évaluations qualitatives ou ordinales ; 2) les évaluations quantitatives ou probabilistes à base de modèles ; 3) les évaluations expérimentales. Chacun de ces types d'évaluation a pour but de produire des mesures. Dans ce paragraphe, nous nous interrogeons sur la définition d'une mesure et sur les différents types de mesures existants avant de présenter les trois types de mesure dans le contexte de la sécurité.

2.1 Sens de la mesure

Dans ce paragraphe, nous nous interrogeons sur le sens de la mesure. "Qu'est-ce qu'une mesure ?" et "Quels types de mesures existent-ils et qu'est-ce qui les différencie ?" sont les questions que nous abordons.

Plusieurs termes peuvent être employés dans le domaine de l'évaluation. La langue française offre deux vocables possibles : "mesure" et "métrique", alors que la langue anglaise en possède trois : "*measure*", "*metric*" et "*measurement*". Cette différence dans le nombre de termes rend plus obscures les différences entre les notions portées par chacun.

Dans un premier temps, examinons les définitions de ces deux termes dans un dictionnaire de la langue française [LAR 02]. Le terme "mesure" existe en tant que nom alors que celui de métrique n'existe qu'en tant qu'adjectif. La mesure est définie comme une action permettant d'évaluer une grandeur d'après son rapport avec une grandeur de même espèce, prise comme unité et comme référence ; grandeur, dimension ainsi évaluée. L'adjectif métrique est défini comme qualifiant ce qui est relatif au mètre. Le dictionnaire de la langue française nous donne des définitions très restrictives qui et indique une utilisation abusive du terme "métrique" dans la langue française, due à l'usage du nom "*metric*" dans la langue anglaise.

Cependant, une définition mathématique de la métrique est donnée dans [ZUS 91] : la métrique est une distance entre deux entités. La plus connue des métriques est la distance d'Euclide (distance minimale entre deux points). Une fonction *f* est qualifiée de métrique si, *x*, *y*, *z* étant des entités :

$$f(x,y) = 0 \text{ pour } x=y ;$$
$$f(x,y) = f(y,x) \quad \forall x, y ;$$
$$f(x,z) \leq f(x,y) + f(y,z) \ \forall x, y, z.$$

Dans [STE 59], le psychologue Stanley Smith Stevens propose une définition de la mesure (*measurement* en anglais) en la définissant comme "l'affectation de nombres à des objets en fonction de règles précises". Cette définition recoupe celle de [ZUS 91], qui définit une mesure (*measure* en anglais) comme une fonction mathématique μ: A $\rightarrow$ B d'un ensemble A d'objets qualifiés d'empiriques (les objets à mesurer) vers un système formel B qui est l'ensemble des valeurs de la mesure (le plus souvent, cet ensemble est $\Re$). On retrouve dans les deux définitions la notion qu'une mesure se rapporte de manière obligatoire à une échelle, notion que nous abordons dans un paragraphe suivant.

Enfin, [ATZ 06] associe le terme *measurement* au procédé visant à acquérir les données nécessaires à l'élaboration d'une mesure désignée par le terme anglais *measure*, qui désigne le résultat final.

Nous avons noté que ces mesures, ou métriques, doivent être rattachées à une échelle pour avoir une signification exploitable. Il existe plusieurs types d'échelle : les échelles nominales, les échelles ordinales, les échelles de ratio et les échelles absolues. Ces échelles sont présentées de la moins contrainte à la plus contrainte. Cependant, plus l'échelle est contrainte, plus elle possède de propriétés et plus les mesures qui s'y réfèrent sont manipulables d'un point de vue mathématique. L'échelle nominale est une énumération non ordonnée : on peut imaginer par exemple classer un jeu de cartes selon les quatre couleurs (pique, cœur, carreau et trèfle). Une telle échelle permet de classer sans autoriser aucune opération entre les classes ainsi constituées. Considérons maintenant ce paquet de cartes non plus selon la couleur de la carte mais sa valeur (roi, dame, valet, 10, etc.). Cette classification est donc basée sur une échelle ordinale : il est possible de comparer les cartes selon leur valeur – par exemple, le roi est plus fort que le valet. L'utilisation d'une échelle d'intervalles ajoute la propriété d'addition à celle de la comparaison. Par exemple, la date ou la température en degré Celsius est basée sur une échelle d'intervalles : l'intervalle entre valeurs est constant mais le zéro est arbitraire, comme l'est l'an 0 dans les différents calendriers existants. L'échelle de ratio apporte la propriété supplémentaire de la division. Par exemple, le poids, l'âge ou la taille sont basés sur des échelles de ratios : l'intervalle entre les unités est constant, mais le rapport entre deux mesures a un sens. Par exemple, un poids de 10 kilogrammes est bien deux fois plus lourd qu'un poids de 5 kilogrammes. A ces quatre échelles distinguées par [STE 59], l'auteur de [ZUS 91] ajoute l'échelle absolue, qui correspond à des quantités sans unités telles que les probabilités. Les auteurs associent à chaque type d'échelles une fonction mathématique (notée *g*) traduisant la transformation d'une mesure à une autre. Ces propriétés sont résumées dans le tableau 1.1.

Echelle nominale	g surjective
Echelle ordinale	g strictement croissante
Echelle d'intervalles	$g(x)=ax+b$, $a>0$
Echelle de ratio	$g(x)=ax$, $a>0$
Echelle absolue	$g(x)=x$

Tableau 1.1 : Propriétés mathématiques associées aux échelles

Une mesure se réfère à une échelle et possède, ou non, une unité. Mais une mesure a aussi et avant tout un objectif : quelle est la réponse que doit fournir l'évaluation ? Une réponse de type OUI/NON peut suffire à répondre à une question telle que "mon système est-il suffisamment protégé pour garantir ma politique de sécurité ?". Mais une réponse plus poussée peut être exigée. Dans le cadre de la sécurité, une évaluation peut répondre à une question du type "Je veux respecter des seuils pour la sécurité de mon système. Quels sont les efforts qui devront être déployés ? ", soit une évaluation d'une distance à objectif. Elle peut également répondre à une question dans le contexte inverse en évaluant les risques à partir d'une situation donnée : "Mon système possède tels moyens de protection, quels risques encourt-il vis-à-vis de la sécurité ?"

De nombreux travaux ont cherché à répondre soit à l'un ou à l'autre de ces deux types de question. Nous tentons de faire ci-dessus un historique et un panorama significatif des travaux d'évaluation pour la sécurité. Dans le paragraphe suivant, nous commençons par les premières approches, motivées par le besoin de sécuriser les systèmes d'informations militaires.

2.2 Premiers besoins, premières approches : l'évaluation qualitative

2.2.1 CC et normes ISO

Les critères d'évaluation représentent une des premières approches déterministes pour l'évaluation de la sécurité. Ces critères permettent une évaluation ordinale d'un système. Les premiers critères apparus sont connus sous le terme de "livre orange" présentés ci-dessous.

2.2.1.1 Le livre orange

Le rapport *Trusted Computer Security Evaluation Criteria* [DOD 85], plus connu sous le nom de TCSEC ou "livre orange", est devenu en 1985 une norme du département de la défense Américaine (DoD). Ce document à longtemps été la référence en matière

d'évaluation de la sécurité des systèmes d'information. Les critères offrent une classification à sept niveaux de sécurité, regroupés en quatre classes A, B, C, D (les niveaux sont A1, B1, B2, B3, C1, C2, D). D est le niveau le plus faible et A1 le niveau le plus élevé. Quatre familles de critères sont définies pour chaque niveau ; chacune traite respectivement d'une politique de sécurité : la politique d'autorisation, de l'audit, de l'assurance et de la documentation. L'évaluation d'un produit consiste à lui attribuer un des sept niveaux de sécurité. Cette attribution n'aboutit que si le produit répond à tous les critères du niveau en question. La politique d'autorisation distingue deux types de politiques, discrétionnaire[7] et obligatoire[8]. Par exemple, un utilisateur sera autorisé à manipuler une information dans le système si le droit en lecture est positionné sur l'information pour lui (contrôle discrétionnaire) et s'il est habilité à la manipuler (contrôle obligatoire). Un système classé D est un système ne présentant aucune caractéristique particulière quant à la sécurité. Les critères permettent l'utilisation d'une politique discrétionnaire pour les niveaux C1 et C2. Les critères exigent l'utilisation d'une politique discrétionnaire et d'une politique obligatoire pour les niveaux B1, B2, et B3. Un système A1 est fonctionnellement équivalent à un système B3 mais est caractérisé par l'utilisation de méthodes formelles de vérification qui permettent d'assurer que les contrôles discrétionnaires et obligatoires employés dans le système sont bien à même de protéger les informations sensibles manipulées par le système. Une documentation détaillée et complète est nécessaire pour démontrer que le système respecte bien les exigences de sécurité à tous les stades de la spécification, de la conception et du codage.

Dans le livre orange, la politique obligatoire imposée est la politique de Bell-LaPadula. Cette politique ne s'intéressant qu'au problème de confidentialité des données, elle ne permet pas de qualifier l'intégrité et la disponibilité de celles-ci. Ce manque a donc suscité la création de nouveaux critères dans d'autres pays. Nous citons ci-dessous l'exemple des ITSEC (*Information Technology Security Evaluation Criteria*) adoptés par la Communauté Européenne, mais d'autres pays tels que le Canada avec les CTCPEC (*Canadian Trusted Computer Product Evaluation Criteria*) [CTC 93] et le Japon [JCS 92] ont également élaboré leurs propres critères d'évaluation. Enfin, les Critères Communs (CC) [CC 96] sont nés de la tentative d'harmonisation des critères canadiens (CTCPEC), européens (ITSEC) et américains (TCSEC).

[7] Dans le cas d'une politique discrétionnaire, les droits d'accès à chaque information sont manipulés librement par le responsable de l'information (généralement, son propriétaire).

[8] Une politique obligatoire impose des règles incontournables qui s'ajoutent aux règles discrétionnaires. Elle suppose que les utilisateurs et les objets aient été étiquetés. On dit que les objets possèdent une classification tandis que les utilisateurs possèdent une habilitation. L'autorisation d'accès est donc basée sur une comparaison entre l'habilitation de l'utilisateur et la classification de l'objet.

2.2.1.2 Les ITSEC

Les ITSEC sont le résultat de l'harmonisation de travaux réalisés au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni [ITS 91]. La différence essentielle que l'on peut noter entre le livre orange et les ITSEC est la distinction entre fonctionnalité et assurance. En effet, les ITSEC définissent un certain nombre de classes de fonctionnalité d'une part et un certain nombre de classes d'assurance d'autre part. Une classe de fonctionnalité décrit les mécanismes qu'un système doit mettre en œuvre pour être évalué à ce niveau de fonctionnalité. Une classe d'assurance permet, elle, de décrire l'ensemble des justifications et preuves qu'un système doit apporter pour montrer qu'il implémente réellement les fonctionnalités qu'il prétend assurer. 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.

Les ITSEC utilisent le terme cible d'évaluation (*Target Of Evaluation* ou TOE). Le contenu d'une TOE comprend : une politique de sécurité, une spécification des fonctions requises dédiées à la sécurité, une définition des mécanismes de sécurité requis et le niveau d'évaluation visé. Les ITSEM (*Information Technology Security Evaluation Manual*) ont été publiés dans leur première version en 1993. Ils décrivent une méthodologie pour mener les évaluations relatives aux critères ITSEC [ITS 93].

2.2.1.3 Les Critères Communs

Les CC, harmonisation des CTCPEC, des ITSEC et des TCSEC, contiennent deux parties bien séparées comme les ITSEC : fonctionnalité et assurance. De même que dans les ITSEC, les CC définissent une cible d'évaluation, appelée TOE, qui désigne le système ou le produit à évaluer, et la cible de sécurité, appelée ST (*Security Target*), qui contient les objectifs de sécurité d'une TOE particulière et qui spécifie les fonctionnalités et les assurances offertes par la TOE afin de remplir ces objectifs. La cible de sécurité pour une TOE représente la base d'entente entre évaluateurs et développeurs. Une cible de sécurité peut contenir les exigences d'un ou de plusieurs profils de protection prédéfinis. Une des différences entre ITSEC et CC réside dans l'existence de ces profils de protection qui avaient auparavant été introduits dans les critères fédéraux des Etats-Unis [FC 92]. Un profil de protection définit un ensemble d'exigences de sécurité et d'objectifs, indépendants d'une quelconque implémentation, pour une catégorie de TOE. L'intérêt des profils de protection est double : 1) un développeur peut inclure dans une cible de sécurité un ou plusieurs profils de protection ; 2) un client désirant utiliser un système ou un produit peut également demander à ce que son système corresponde à un profil de protection particulier, ceci lui évitant de donner une liste exhaustive de fonctionnalités et assurances qu'il exige du système ou du produit. Une partie importante des CC est donc consacrée à la présentation détaillée de profils de protection prédéfinis. Cette notion de profil de protection représente la volonté américaine qui consiste à préférer évaluer des systèmes qui entrent dans le cadre de profils connus. Le cas d'un système ne cadrant pas exactement avec un profil connu conduit simplement à l'élaboration d'un nouveau profil. La tendance européenne serait plutôt de définir systématiquement une TOE et ses exigences de sécurité et d'évaluer cette TOE sans s'appuyer nécessairement sur des profils prédéfinis. Les CC tentent de réaliser un compromis entre ces deux positions.

2.2.1.4 Les normes ISO

La norme ISO 17799 de sécurité a été publiée dans sa première version en 2000. Une seconde version a été publiée en 2005. Ce document s'adresse tout d'abord aux

responsables de la sécurité et aux directions informatiques quel que soit le secteur d'activité. Il a pour but de définir des objectifs et des recommandations à propos de la sécurité des systèmes d'information [CLU 03]. Un premier "code de bonnes pratiques", défini par plusieurs grandes entreprises britanniques (Shell, British Telecom, Midland Bank et Mark&Spencer), sur la base de leur expérience, est à l'origine de cette norme. Ce premier document est publié en 1993 par le *British Standard Institute*. Plusieurs versions de ce document se succèdent jusqu'au BS 7799:1999, dont la première partie est standardisée en 2000 pour devenir ISO 17799:2000. Elle est révisée par la suite et publiée à nouveau en 2005 sous le nom de ISO 17799:2005. Elle fournit la matière nécessaire à la bonne gestion de la sécurité au sein de l'entreprise en abordant les thématiques suivantes [ISO 00] :

- la politique de sécurité ;
- l'organisation de la sécurité ;
- la classification et le contrôle des biens ;
- la sécurité et les ressources humaines ;
- la sécurité physique ;
- la gestion des opérations et des communications ;
- les contrôles d'accès ;
- le développement et la maintenance des systèmes ;
- la gestion de la continuité d'activité ;
- la conformité et la réglementation interne et externe.

Face aux problèmes de sécurité informatique, de nombreuses entreprises cherchent à obtenir la certification relative à cette norme : en 2005, 25% des entreprises interrogées pour le *Security Survey* d'Ernst & Young [ERN 05] étaient certifiées ISO 17799 et 30% envisageaient de l'obtenir.

Dans un nouveau système de numérotation mis en place récemment, cette norme a rejoint la famille des normes ISO 27000 [ISO 08] (dont la première norme est présentée dans le paragraphe suivant) en étant renommée ISO 27002.

La norme ISO 27001 est issue de la seconde partie de la norme britannique BS 7799 présentée dans le paragraphe précédente. Sa première et unique version sort en 2005. La norme ISO 27001 s'intéresse au Service Général du Système d'Information en permettant des audits internes et une amélioration du système de gestion de la sécurité. En appliquant en permanence des méthodes d'analyse de risque, incorporées dans le modèle de processus PDCA (Planifier – Faire – Vérifier – Agir), elle permet un meilleur cadre de gestion qui rend possible les pratiques de contrôle de la norme ISO 17799:2005.

Outre la norme ISO 27001 décrite ci-dessus et la norme 17799:2005 bientôt rebaptisée 27002, c'est au moins quatre autres normes qui sont à l'étude. Les contenus de ces normes sont résumés dans le tableau 1.2 ci-dessous :

ISO 27001	Spécifications pour les systèmes de gestion de la sécurité de l'information, ancien BS7799-2.
ISO 27002	Actuelle référence du standard ISO 17799:2005.
ISO 27003	Futur standard offrant un guide d'implémentation d'un système de gestion de la sécurité de l'information.
ISO 27004	Futur standard sur les mesures et métriques destinées à la gestion de la sécurité des systèmes d'information.
ISO 27005	Futur standard sur la gestion des risques liés à la sécurité de l'information.
ISO 27006	Futur standard donnant des directives pour l'accréditation des organisations offrant la certification des systèmes de gestion de la sécurité de l'information.

Tableau 1.2 : La famille des normes ISO 27000

D'autres normes existent déjà ou sont destinées à être publiées (ISO 24760). La thématique de chacune d'entre elles sera par la suite reprise dans un standard de la famille des ISO 27000. La norme ISO 13335 (norme en cinq parties publiées entre 1998 et 2004) concerne la gestion de la sécurité des technologies de l'information et de la communication. Une partie de ce standard se retrouvera dans l'ISO 27005. La norme ISO 17021 (la première partie a été publiée en 2006, la seconde est à paraître) contient les principes et les spécifications pour la compétence, l'objectivité et l'impartialité des audits et des certifications des systèmes de gestion. Cette thématique sera retrouvée dans l'ISO 27006. Enfin, la norme ISO 24760 est encore à paraître mais sa thématique sur les techniques pour la sécurité des technologies de l'information est déjà évoquée dans la norme ISO 27002. La norme ISO 15408 est apparue en 1996 et contient les Critères Communs que nous avons décrits précédemment. On peut évoquer également la famille des standards ISO 9000 qui sont des normes relatives à la qualité. Certaines abordent la qualité du logiciel, comme l'ISO 9126:2001.

2.2.2 Les méthodes d'analyses de risques

Il existe de nombreuses méthodes d'analyses de risques, et ce pour de nombreux domaines. Outre le domaine de la sécurité des systèmes d'information que nous abordons ici, les domaines de la santé, de l'alimentaire ou encore les domaines industriels ont recours à des analyses de risques. Bien que ces domaines soient très variés, la démarche générale d'une analyse de risque 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.

On qualifie de risque un événement redouté auquel sera attribuée une fréquence de réalisation et une conséquence [DAC 94]. Les risques sont donc évalués à partir d'une estimation de la fréquence de réalisation des menaces et des conséquences [ABG 04].

En France, il existe plusieurs méthodes d'analyse de risques dédiées aux systèmes d'information. MARION (Méthodologie d'Analyse de Risques Informatiques Orientés par Niveaux) a été développée en 1980 par l'Assemblée Plénière des Sociétés d'Assurance Incendie et Risque Divers. Sa dernière mise à jour date de 1998. MELISA a été développée pour la Délégation Générale de l'Armement, mais MELISA et MARION étant relativement anciennes, ces deux méthodes ne sont plus très utilisées.

Plus récemment, d'autres méthodes ont été développées. MEHARI (MEthode Harmonisée d'Analyse de Risques), développée par le CLUb de la Sécurité des systèmes d'Information Français en 1995. Cette méthode a été mise à jour en février 2007. La méthode EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité), développée par la Direction Centrale pour la Sécurité des Systèmes d'Information, a vu le jour en 1995.

Enfin, plusieurs méthodes d'analyse de risques ont été développées à l'étranger ou dans le secteur français privé. Parmi elles, on retrouve Octave (*Operationally Critical*

Threat, Asset, and Vulnerability Evaluation) développée en 1999 par le *Software Engineering Institute* (USA) ou encore SCORE.

Nombre de ces méthodes, comme MEHARI[9], fournissent des mesures telles que l'impact global d'une attaque. Cependant, ces mesures, sont tantôt appelées quantitatives, tantôt qualitatives. Dans le paragraphe suivant, nous nous interrogeons sur cette terminologie.

2.2.3 Du qualitatif au quantitatif

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. Cependant, les évaluations fournies par les normes ISO et les méthodes d'analyse de risques découlent d'un processus long ne permettant 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. Pour ces raisons, des études cherchant à étudier la quantification de la sécurité en mode opérationnel ont vu le jour. Cependant, il faut se poser la question suivante : où se situe la frontière entre mesure qualitative et mesure quantitative ? La frontière semble à ce jour toujours un peu floue. En effet, dans [LAP 95], il est défini que, dans le contexte de la sûreté de fonctionnement, une évaluation ordinale était qualifiée de qualitative et une évaluation probabiliste de quantitative, alors que les deux types d'évaluation sont qualifiées de quantitatives dans le contexte de la sécurité. Cependant, les mesures issues des critères ou des normes ISO sont pourtant citées comme étant des mesures qualitatives par de nombreuses approches [DAC 94, ALA 07, WAN 05]. Dans [JAQ 07], l'auteur va même plus loin en affirmant que ce ne sont pas des mesures. La frontière entre les notions de mesure qualitative et mesure quantitative. Associer ces notions à des échelles de mesure, présentées dans la partie 2.1, permet de ne pas laisser d'ambiguïté.

Ainsi, dans ce mémoire, nous nous baserons sur la définition simple qui est qu'une mesure qualitative est une mesure dont les valeurs traduisent l'appartenance à une classe. On peut associer, comme le fait [LAP 95] l'échelle ordinale aux mesures qualitatives. Les mesures se référant aux échelles d'intervalles, de ratio et absolue sont donc des mesures quantitatives. L'échelle nominale, ne permettant pas l'opération de comparaison, n'apparaît pas suffisamment contrainte pour être prise en compte dans notre approche.

[9] www.clusif.asso.fr

Mais que peut-on mesurer quantitativement lorsqu'on évalue la sécurité ? En faisant le parallèle avec le domaine de la sûreté de fonctionnement, une quantité qui apparaît comme étant mesurable est la durée. Les mesures de MTTF (*Mean Time To Failure* en anglais) et de MTBF (*Mean Time Between Failure* en anglais) font partie des plus utilisées dans le domaine de la sûreté de fonctionnement. Ces mesures sont strictement quantitatives et basées sur une échelle absolue. Dans [LIT 93], l'auteur évoque la quantification de l'effort de l'attaquant en mettant en parallèle la notion de temps dans le cadre de la sûreté de fonctionnement et d'effort à fournir par l'attaquant dans le cadre de la sécurité. Cependant, la compétence de l'attaquant représente dans ce contexte une notion importante. L'auteur ne précise pas comment il serait possible de l'évaluer de façon quantitative. Mais des mesures peuvent également valuer l'efficacité des mécanismes de protection comme par exemple les IDS (*Intrusion Detection Systems*), dont plusieurs aspects peuvent êtres évalués comme le ratio de détection ou le ratio de fausses alarmes.

Dans le paragraphe suivant, nous présentons des approches nées de ce besoin de quantification.

2.3 Méthodes d'évaluation en mode opérationnel

Il existe dans la littérature deux types d'évaluation conduisant à des mesures quantitatives pour la sécurité : les mesures issues de l'expérimentation et les mesures issues d'évaluations basées sur des modèles. Dans ce paragraphe, nous présentons des travaux issus des deux approches.

2.3.1 Les évaluations expérimentales

Les évaluations expérimentales sont basées sur des données observées. Ces données observées peuvent être de deux natures : 1) les données issues d'une expérimentation sensée reproduire le monde réel et 2) les données issues d'observations réelles. Dans [JON 97], l'auteur a conduit une expérience sur 24 étudiants en leur proposant, en binômes, de mener des attaques sur un système d'information. Les observations qui résultent de cette expérience ont produit des mesures quantitatives telles que le MTTB (*Mean Time To Breach*), ainsi qu'un modèle qualitatif d'évolution des connaissances de l'attaquant. Cependant, il est à noter que l'échantillon peu important et peu homogène de l'expérience réduit la portée de ses résultats.

La seconde technique d'évaluation expérimentale consiste à collecter des données issues d'activités réelles. Pour être ensuite utilisables, ces données doivent être validées en triant les données réelles des données erronées, et en considérant le manque d'exhaustivité du jeu de données. Ces erreurs peuvent survenir d'un dysfonctionnement du mécanisme d'observation. Cette étape est essentielle pour permettre une exploitation de ces données et une production de résultats qui ne soient pas biaisées.

Dans le cadre de la sécurité, on peut citer les processus de collecte de données grâce aux pots de miel. Un pot de miel est un système informatique volontairement vulnérable à une ou plusieurs failles et connecté à un réseau tel qu'Internet de façon à être accessible aux attaquants [ALA 07]. Un tel système n'enregistre donc aucune activité si ce n'est des tentatives d'attaque ou des tentatives de connexion erronées et accidentelles. Dans [ALA 07], les données analysées furent collectées pendant plus de trois ans dans le cadre du projet Leurré.com [ALA 05]. Leur analyse a permis, entre autres, de modéliser la durée des intervalles de temps entre attaques.

Si ces travaux sont partis de données expérimentales pour obtenir un modèle valide, le procédé inverse est également utilisé en basant l'évaluation de la sécurité sur des modèles. Ces travaux sont présentés dans le paragraphe suivant.

2.3.2 Les évaluations basées sur des modèles

Ce paragraphe présente plusieurs des approches existantes pour l'évaluation des systèmes d'information. Il ne s'agit pas de faire une liste exhaustive de ce qui existe mais plutôt un panorama des principales approches et du cadre d'étude choisi dans chaque cas. Nous commençons par l'une des premières approches qui a été développée au LAAS, l'approche du graphe des privilèges.

2.3.2.1 Approche du LAAS : graphe des privilèges

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 au LAAS [DAC 94, DAC 96, ORT 98]. 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.

L'élaboration du graphe des privilèges à partir des vulnérabilités d'un système se situe durant la première étape de cette démarche. Un privilège se définit comme un ensemble de droits qu'un utilisateur, ou qu'un groupe d'utilisateurs, peut posséder sur un objet. Chaque nœud du graphe représente un ensemble de privilèges. Un arc reliant deux nœuds du graphe indique qu'un utilisateur possédant le premier ensemble de privilèges peut acquérir le second par exploitation d'une vulnérabilité. Les vulnérabilités exploitées peuvent êtres dues à des faiblesses du système mais peuvent aussi représenter des mécanismes propres et nécessaires au fonctionnement du système, et pour faciliter le travail coopératif entre les utilisateurs.

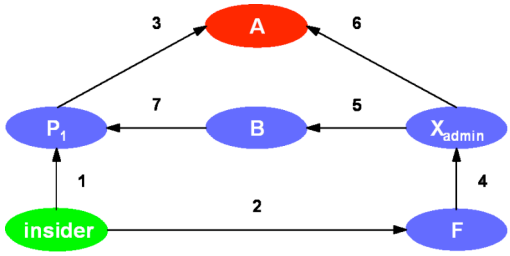


Figure 1.3 : Exemple de graphe de privilèges

1	"peut deviner le mot de passe de"
2	"est présent dans le fichier .rhost"
3	"peut modifier un répertoire dans la liste $PATH de"
4	"peut modifier le fichier .xinitrc de"
5	"peut réaliser une attaque par le biais du courrier électronique"
6	"est un sur-ensemble des privilèges de"
7	"peut modifier l'exécution d'un programme dont le setuid bit est positionné et qui appartient à"

Tableau 1.3 : Vulnérabilités présentes dans le système

La figure 1.3 présente un exemple de graphe des privilèges que l'on pourrait construire à partir de l'observation d'un système Unix comportant plusieurs vulnérabilités. Chaque numéro sur les arcs représente la méthode utilisée pour l'acquisition de l'ensemble de privilèges ciblé. Des exemples de vulnérabilités correspondant à la figure 1.3, en considérant le cas d'un système Unix, sont présentés dans le tableau 1.3 ci-dessus. Dans l'exemple de la figure 1.3, les nœuds représentent les ensembles de privilèges relatifs à un utilisateur (*A* : ensemble de privilèges de l'utilisateur A) ou à un ensemble d'utilisateurs (par exemple, un groupe dans Unix). Les nœuds (*A*,*B*,*F*) représentent des privilèges d'utilisateurs et (X_{admin},*P*) des privilèges de groupes d'utilisateurs. Le nœud *insider* représente les privilèges minimaux dont dispose tout utilisateur du système (par exemple, le privilège de se connecter ou de changer son mot de passe).

Dans un graphe des privilèges, on peut identifier des nœuds que nous appellerons "cible" qui correspondent à des privilèges que l'on souhaite protéger (par exemple, les privilèges du super-utilisateur). Ces nœuds représentent les objectifs de sécurité du système. Par ailleurs, on peut identifier des nœuds appelés "attaquant" qui représentent les privilèges d'attaquants potentiels. Tous les chemins entre un nœud attaquant (par exemple "*insider*") et un nœud cible (par exemple "*A*") sont des possibilités pour l'attaquant de mettre en défaut la politique de sécurité du système. De tels chemins existent dans la plupart des systèmes même s'ils ne sont pas tous facilement exploitables. Par exemple, tous les mots de passe peuvent être devinés : certains sont faciles à trouver par des outils parce qu'ils figurent dans un dictionnaire, alors que d'autres nécessitent plus d'effort et de temps. Ceci est vrai pour toutes les classes de vulnérabilités : certaines sont facilement exploitables par un attaquant alors que d'autres nécessitent beaucoup de compétence, ténacité ou chance.

Par ailleurs, une pondération des arcs a été suggérée pour représenter l'effort moyen nécessaire pour réussir l'attaque correspondante. Pour cela, une classification qualitative sur quatre niveaux a été proposée, comme montré dans le tableau 1.4.

Niveau 4	Attaques sophistiquées ; l'attaquant doit être près du système
Niveau 3	Attaques sophistiquées, connues théoriquement mais non recensées
Niveau 2	Attaques connues mais rarement recensées
Niveau 1	Attaques connues, fréquemment utilisées

Tableau 1.4 : Classification des attaques par effort

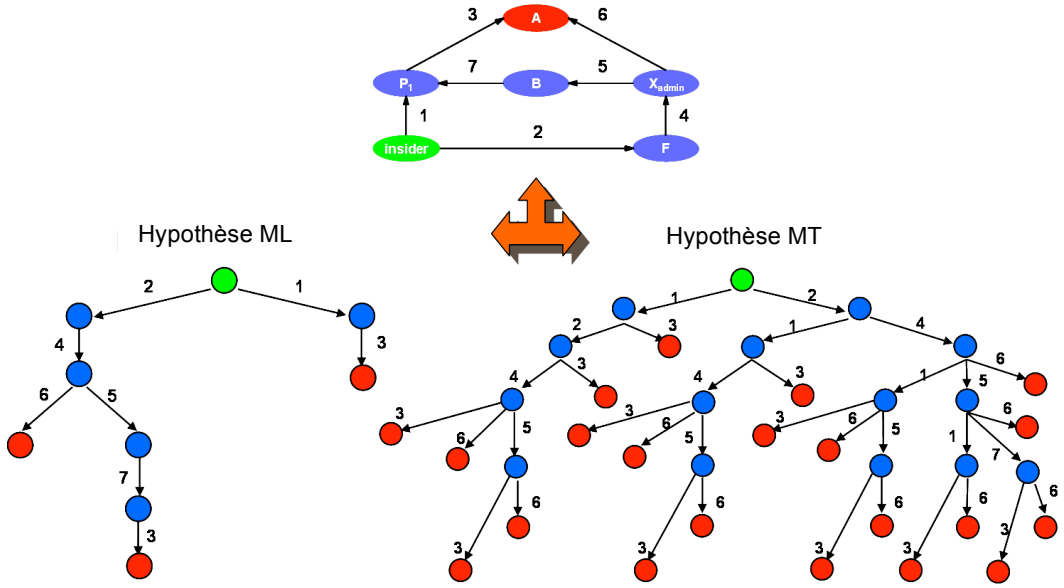


Figure 1.4 : Déploiement du graphe des privilèges en graphes d'attaque

Comme nous l'avons vu plus haut, la seconde étape de l'approche est l'obtention de graphes d'attaque à partir d'un graphe de privilèges. Certaines hypothèses sur le comportement d'un attaquant potentiel sont nécessaires pour identifier les scénarios d'attaque permettant de construire le graphe d'attaque. A titre d'illustration, la figure 1.4 donne les graphes d'attaque obtenus à partir du graphe des privilèges de la figure 1.3 en considérant deux hypothèses différentes du comportement de l'attaquant, notées MT (Mémoire Totale) et ML (Mémoire Locale), respectivement. Pour cet exemple, nous rappelons que *A* est la cible et *insider* (noté *I*) est l'attaquant. Avec l'hypothèse MT, à chaque étape du processus d'attaque, l'attaquant peut choisir une attaque parmi toutes celles qu'il a identifiées durant les étapes précédentes et qui n'ont pas abouti, c'est-à-dire qu'il essaie de tirer avantage de tous les privilèges acquis, même ceux acquis en exploitant une vulnérabilité d'un chemin différent. Par contre, avec l'hypothèse ML, seules les attaques réalisables avec les nouveaux privilèges acquis à partir du nœud du graphe qu'il vient d'atteindre sont considérées. Dans l'exemple de la figure 1.4, dans l'hypothèse ML, après avoir exploité la vulnérabilité 2, l'attaquant ne "se souvient" plus qu'il a la possibilité d'exploiter la vulnérabilité 1. Par contre avec l'hypothèse MT, la vulnérabilité 1 sera prise en compte dans les possibilités d'attaque tentées par l'attaquant.

La troisième étape de la démarche consiste en l'obtention de mesures depuis les graphes d'attaques. Leur transformation en chaînes de Markov permet de déduire pour un couple (cible-attaquant) la mesure *Mean Effort To Failure* en considérant l'hypothèse ML ou MT, le plus court chemin. De plus, cette mesure tient compte de l'influence des "impasses", chemins qui ne permettent pas à l'attaquant d'arriver à ses fins. Cette approche permet l'identification de chemins critiques dans le système. De plus, le graphe des privilèges pouvant évoluer selon le comportement et l'utilisation du système au jour le jour, un écart lors d'un calcul régulier de la mesure permet de mettre en évidence une baisse du niveau de sécurité.

2.3.2.2 Modèles de scénarios : arbres et graphes d'attaque

Depuis les travaux menés au LAAS sur l'évaluation quantitative de la sécurité et la mise au point du graphe des privilèges, d'autres travaux ont été menés pour la représentation des processus d'attaque. Le principe reste le même : la mise en évidence des différents scénarios qui pourraient être suivis par l'attaquant pour atteindre son objectif. Par exemple, dans l'approche [SHE 04], chaque état du graphe ne représente pas seulement l'ensemble des privilèges de l'attaquant mais 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 doit dans ce cas-là être réduit pour pouvoir être exploité. Des approches pour générer et réduire ces graphes sont présentées dans [JHA 02, SWI 01, SHE 04, NOE 04].

D'autres travaux se sont focalisés sur la représentation des processus d'attaque par des arbres d'attaque [RIC 02]. Les arbres d'attaques constituent une adaptation des arbres de fautes qui 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. Contrairement aux graphes d'attaque présentés sur la figure 1.4, les arbres d'attaque ont pour racine le but de l'attaquant, représenté par un losange. L'arbre présente ensuite les différentes façons d'arriver à ce but : une « généalogie » de l'attaque [RIC 02]. Les rectangles de ce graphe sont les sous-objectifs à atteindre avant d'arriver au but final de l'attaquant. L'arbre d'attaque met en relief les différents moyens pour l'attaquant d'atteindre ses objectifs.

Un exemple d'arbre d'attaque est donné dans la figure 1.5 : l'attaquant veut accéder au coffre-fort sans avoir à le forcer, en l'ouvrant avec la combinaison.

Dans cet exemple, on voit que pour pouvoir ouvrir le coffre-fort, l'attaquant doit découvrir la combinaison. La découverte de la combinaison devient son premier sous objectif. Pour l'atteindre, deux possibilités s'offrent à lui : soit trouver un document contenant la combinaison, soit obtenir la combinaison de quelqu'un qui la connaît. Le choix d'une alternative définira des sous-objectifs de niveau inférieur et ainsi de suite.

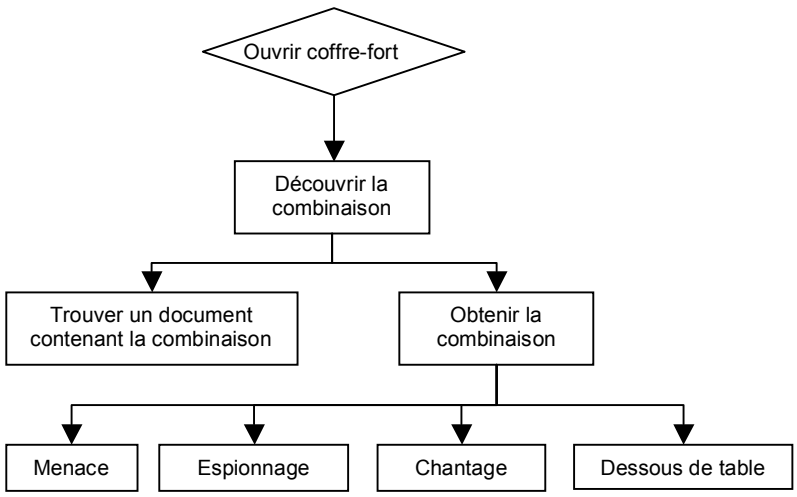


Figure 1.5 : Exemple d'arbre d'attaque

L'approche présentée dans [BAL 06] utilise ce formalisme. Elle produit une mesure de risque en utilisant une méthodologie en deux étapes : 1) la première consiste en l'élaboration de l'arbre d'attaque, chaque noeud 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 ; 2) 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.

Dans [WAN 97], le système est décomposé sous forme d'un arbre jusqu'à obtenir des composants élémentaires indépendants. Pour chacun de ces composants sont évaluées la probabilité d'exploitation avec succès de la vulnérabilité de ce composant et la probabilité que cette vulnérabilité soit choisie. La probabilité d'exploitation avec succès du système est donc obtenue en remontant l'arbre établi. Une démarche similaire a été adoptée dans [WAL 06] où la décomposition en éléments est effectuée en adoptant une analogie à la défense des châteaux forts médiévaux.

Notons qu'un arbre d'attaque n'est pas simple à élaborer [SCH 99], car il n'y a pas de technique automatisée pour produire formalisme, ce qui rend fastidieuse la mise au point de la mesure.

2.3.2.3 Evaluation des processus d'attaque et de protection

L'approche décrite dans [MCQ 06] ne modélise pas le système mais le comportement de l'attaquant face au système. Cette approche utilise une description de trois processus d'attaque génériques sous forme d'organigrammes de programmation. Pour chacun des processus 1 et 2, une des conditions suivantes est vérifiée : 1) pour le processus 1, il existe au moins une vulnérabilité connue dont l'exploitation permet d'obtenir les privilèges voulus et l'attaquant connaît au moins une attaque disponible qui exploiterait avec succès cette vulnérabilité ; 2) pour le processus 2, il existe au moins une vulnérabilité connue dont l'exploitation permet d'obtenir les privilèges voulus ; l'attaquant ne connaît pas d'attaque disponible et recherche donc une attaque qui exploiterait avec succès cette vulnérabilité. Dans le processus 3, l'attaquant cherche à identifier de façon permanente de nouvelles vulnérabilités puis des attaques exploitant celles-ci. Les processus 1 et 2 sont exclusifs et concernent l'exploitation des vulnérabilités déjà connues. Le processus 2 est exécuté seulement si le processus 1 échoue et si les conditions de départ de ce processus ne sont plus valides. Le processus 3 est en exécution permanente et parallèle aux processus 1 et 2. La mesure "Time To Compromise" résultante dépend donc de la probabilité d'occurrence de ces processus et du temps nécessaire au succès de l'attaquant pour chacun d'entre eux. Cette mesure nécessite de connaître le nombre de vulnérabilités présentes dans le système étudié et donne des mesures dépendant d'un seul attaquant.

Dans [IRV 99], les auteurs n'évaluent pas le processus d'attaque mais le coût du moyen de protection à utiliser selon l'attaque et l'attribut de sécurité mis en danger. Ces mesures sont évaluées par un coût matériel : occupation de la bande passante en octet par seconde, en octet à stocker, en puissance du processeur consommée, etc.

2.3.2.4 Evaluation issue de la théorie des jeux

Plusieurs modèles utilisent la théorie des jeux afin d'envisager les différents scénarios entre l'attaquant et le système de défense du système d'information.

Dans [SAL 05], les auteurs proposent de prendre en considération la nécessité pour l'attaquant de se montrer discret et de ne pas mener des attaques facilement discernables par l'administrateur système. Une fois ce critère évalué et pris en compte, les probabilités de décision sont calculées grâce à un raisonnement de type Minimax[10]. Cette approche laisse donc supposer que l'attaquant à une connaissance approfondie du système, tant au niveau des vulnérabilités que de la qualité des ripostes du système face à l'exploitation de ces vulnérabilités. Ces aspects sont retrouvés dans [SAL 06] qui ajoute à son modèle probabiliste la probabilité de décision de l'attaquant : une variable contient la probabilité que l'attaquant choisisse une action particulière.

Cette approche par la théorie des jeux est donc utilisée pour la prise de décision d'une réponse immédiate du système comme dans [LIU 05] où le jeu considéré est un jeu dont la somme des gains est nulle : les gains obtenus par un (ou plusieurs) des joueurs sont égaux à la somme des pertes des autres joueurs. De plus, le modèle prend en compte un trait de caractère du comportement de son attaquant : celui-ci peut être rationnel (conscient des risques et visant à minimiser le coût de son attaque), ou bien irrationnel (non conscient des dangers qu'il encoure et cherchant uniquement à réussir son attaque).

Ces approches permettent d'attribuer des points à l'attaquant ou au système en fonction des attaques et des réponses du système, mesure basée sur une échelle d'intervalles.

Un autre modèle a été proposé dans [MAD 02, MAD 02a]. Les auteurs ont construit le modèle de l'*attack-response graph*. Ce modèle a pour ambition de prendre en compte la riposte du système face aux attaques : un exemple type est celui des systèmes tolérants aux intrusions. Considérant un graphe d'attaque représentant les divers scénarios possibles que pourraient adopter l'attaquant, si l'exploitation d'une vulnérabilité est découverte par le système, les différents chemins sont suivis pour tenter d'anticiper et d'enrayer l'attaque. Ce processus est dérivable en une chaîne de Markov, mettant en évidence les taux de transitions d'une vulnérabilité à l'autre mais aussi les arcs marqués des taux de réparation et permettant ainsi de calculer les différentes probabilités de compromission du système.

[10] Algorithme mis au point en 1928 par John Von Neumann, il consiste à passer en revue toutes les possibilités pour un nombre limité de coups et à leur assigner une valeur prenant en compte les bénéfices pour le joueur et pour son adversaire. Le meilleur choix étant alors celui qui maximise les bénéfices du joueur tout en minimisant ceux de son adversaire.

2.3.2.5 Mesures de propagation des vers

En 1988, l'un des premiers vers, mis au point par Robert Morris, une expérience ayant échappé au contrôle de son créateur, selon lui, contamina 6000 machines. Depuis, des centaines de vers ont vu le jour, provoquant plus ou moins de dégâts. Le ver Melissa, apparu en 1999, fit le tour du monde en moins de deux jours. ILOVEYOU, en 2000, infecta 45 millions de machines en un mois. Mais le record est battu en 2002, avec l'apparition du vers Slammer qui infecta 12 000 machines dans les dix minutes suivant les premières infections [MOO 03]. Au total, Slammer toucha 245 000 machines d'après *Internet Security System* et affecta cinq des treize serveurs DNS "racines". Après lui sont apparus, pour les plus remarquables, Blaster/Lovesan (2003, 500 000 machines infectées en quelques heures), Sasser (2004) ou MyDoom (2004) dont les infections ont repris récemment. Les vers de la famille Netsky, apparus en 2004, font toujours beaucoup de victimes aujourd'hui[11].

Face à cette menace, plusieurs modèles ont été mis au point pour analyser la propagation des vers en s'inspirant des trois modèles existants pour la propagation d'épidémies décrits dans [HET 89] : SIS, SIR et SIRS. Le modèle SIS est un modèle qui considère une population totale constante, une population susceptible d'être infectée et une population infectée. Ces deux dernières évoluent dans le temps en considérant un taux d'infection et un taux de guérison fixes. Le modèle SIR prend en considération une population globale non constante pour intégrer les cas de décès par infection, une population susceptible d'être infectée, une population infectée et une population immunisée. Le modèle SIRS prend à la fois en considération les cas de guérison et les cas d'immunité. Enfin, le modèle KS est dérivé des précédents, en ajoutant un signal appelé *kill signal*. Ce signal consiste à prévenir une partie des hotes non vulnérables lorsqu'un hote est infecté, comme un utilisateur dont le système est infecté alarmera son entourage à propos de ce danger. Ces modèles sont évoqués dans [KIM 04, OKA 05]. Dans [KIM 04], le modèle SIR est étendu en ajoutant un taux de perte d'immunité. Ces modèles sont à l'origine des modèles épidémiologiques déterministes. Cependant, dans [OKA 05], le modèle SIS est rapproché du processus markovien de naissance et de mort pour produire une approche probabiliste. L'auteur produit ainsi des mesures telles que la probabilité de contamination totale ou de rémission totale du système.

2.4 Conclusion

Les mesures présentées dans ce paragraphe proposent plusieurs techniques d'approche et plusieurs points de vue. Le premier point de vue que nous pouvons adopter

[11] www.viruslist.com

est le type de mesure produite et les unités utilisées : certaines mesures ont une unité de temps comme la mesure *Time To Compromise* décrite dans l'approche de [MCQ 06]. De nombreuses approches ont suivi la tendance de [LIT 93] en produisant des mesures d'effort telles que le *Mean Effort to Failure* produit par le graphe des privilèges [DAC 94]. Enfin, d'autres mesures sans unité que ces mesures d'effort existent telles que celles produites par la théorie des jeux et qui consistent en un nombre de points gagnés ou perdus par l'attaquant ou le système. Ces mesures sont basées sur une échelle ordinale et contiennent moins d'information.

Ce panorama des mesures pour la sécurité permet de nous faire une idée sur les attributs à considérer pour les mesures que nous comptons produire. En effet, le choix d'une échelle de ratio ou une échelle absolue semble important. Les valeurs issues de mesures basées sur des échelles moins contraignantes, telles que l'échelle d'intervalles, ne permettent pas de mettre en évidence une distance à objectif par leur origine arbitraire. De plus, et malgré les études menées dans cette direction, nous décidons de ne pas nous orienter vers des mesures d'effort. En premier lieu, ce choix est motivé par la vision subjective que peut avoir cette notion. L'effort dépend également des compétences. Bien que [MCQ 06] quantifie la compétence de l'attaquant grâce au nombre de vulnérabilités que ce dernier sait exploiter, il apparaît que la difficulté d'exploitation de la vulnérabilité est un facteur non négligeable à prendre en considération. Un attaquant novice connaîtra les attaques issues de recettes dites de cuisine. Un attaquant expert les connaîtra également mais en connaîtra d'autres beaucoup plus subtiles. La différence en nombre de vulnérabilités peut ne pas être importante tandis que la différence d'expérience et de compétence nécessaire pour exploiter ces vulnérabilités peut s'avérer très grande. De plus, l'expert saura improviser face à une vulnérabilité n'ayant pas de kit d'exploitation disponible à la différence du novice. Il est donc difficile d'envisager la compétence d'un attaquant comme étant proportionnelle au nombre d'attaques qu'il connaît. En second lieu, l'effort de l'attaquant peut aussi être considérablement modifié par la puissance matérielle et logicielle que celui-ci a à sa disposition. Une attaque par force brute ne demande pas beaucoup de compétence ni d'effort. Elle peut en revanche se révéler extrêmement rapide et efficace si l'attaquant possède la puissance de calcul adéquate. L'effort dans ce contexte est donc une unité de mesure qui se concentre sur l'attaquant face au système. Cette mesure d'effort semble donc pouvoir être reliée à la mesure de durée.

- Nous notons que ce type d'évaluation permet une mesure des risques encourus par le système en considérant une situation précise. Pourtant, complément à ce point de vue, il nous semble nécessaire d'en distinguer un second : l'évaluation d'une distance à un objectif. Ces deux objectifs de mesures nous apparaissent complémentaires et il nous semble important d'en tenir compte.

De plus, nous avons vu dans la présentation des mesures quantitatives précédentes que chaque approche prenait en considération certains aspects du système et certains traits de son environnement. Les approches décrites dans [MCQ 06, DAC 94, WAN 97] prennent en considération un élément incontournable du processus d'attaque : le comportement de la population des attaquants, en se focalisant sur l'étendue de leurs compétences et leur comportement d'attaque [MCQ 06], ou leur stratégie [DAC 94, WAN 97]. Le comportement de l'attaquant est donc un élément clé dans le processus d'exploitation d'une vulnérabilité. Mais est-il le seul élément de l'environnement ayant une influence sur le processus d'attaque ?

Dans le prochain chapitre de ce mémoire, nous étudions les éléments ayant une influence sur le processus d'exploitation d'une vulnérabilité : le cycle de vie de la vulnérabilité, le comportement de la population des attaquants - et comment ceux-ci

évoluent dans le temps. Notre objectif est de pouvoir caractériser ces facteurs environnementaux afin de pouvoir quantifier la sécurité de notre système vis-à-vis du processus d'exploitation d'une vulnérabilité.

Chapitre 2 : Caractérisation de l'environnement du système et présentation de l'approche

Dans le chapitre précédent, nous avons fait le tour d'horizon des approches pour l'évaluation de la sécurité. Nous avons évoqué les limites d'une approche qualitative et les besoins qui ont motivé une évaluation quantitative de la sécurité. Nous y avons distingué deux objectifs : pouvoir évaluer une distance à un objectif de sécurité par d'autres outils mathématiques que la comparaison offerte par une échelle ordonnée, ou bien quantifier les risques dans un contexte donné. Nous avons présenté plusieurs techniques d'évaluation quantitative et plusieurs approches s'inscrivant dans cette démarche. Cet état de l'art nous a permis de mettre en évidence un point de vue non exploré que nous trouvions intéressant d'approfondir : étudier un moyen d'évaluer quantitativement la sécurité en considérant un environnement du système plus complexe et en tenant compte des conséquences de l'évolution de cet environnement sur le système.

Dans ce second chapitre, nous présentons le cadre de notre approche. Dans une première partie, nous faisons le point sur les vulnérabilités que nous allons considérer dans notre approche et justifions cette sélection. Dans une seconde partie, nous étudions le cycle de vie d'une vulnérabilité. La troisième partie analyse le comportement de la population d'attaquants. La quatrième partie étudie le comportement de l'administrateur du système. Enfin, une cinquième partie conclue ce chapitre en présentant une première mesure simple - la mesure de la zone de grand risque - qui illustre l'influence des facteurs environnementaux que nous présentons dans ce mémoire.

1 Cadre de l'approche : les types de vulnérabilités

1.1 Définition de la vulnérabilité : rappels et analyse

Dans le chapitre précédent, nous avons étudié la définition même d'une vulnérabilité. Nous avons examiné plusieurs définitions et établi qu'elles convergeaient toutes vers le même concept. Nous avons donc adopté la définition donnée dans le projet MAFTIA [MAF 03], que nous rappelons ici : "une vulnérabilité est une faute accidentelle, ou intentionnelle, malveillante ou non, dans les spécifications, la conception ou la configuration du système, ou dans la manière dont il est utilisé". Est-il possible de tenir compte globalement des différents types de fautes possibles contenus dans la définition de la vulnérabilité ou faut-il les étudier de manière distincte ?

Pour répondre à cette question, il ne faut pas oublier le cadre d'étude que nous avons choisi. Notre objectif est de fournir des mesures quantitatives pour la sécurité considérant les facteurs environnementaux ayant une influence significative environnement du système que nous puissions caractériser de manière quantitative.

Notre première distinction se fait au niveau de la phase de création ou d'occurrence de la faute. Une faute peut être créée durant la phase de développement – que l'on peut

associer aux phases de spécification et conception dans la définition de [MAF 03] – ou la phase opérationnelle du système – que l'on peut associer à la configuration du système ou aux modes d'utilisation.

Considérons en premier lieu les vulnérabilités liées à la phase de développement du système. Ces vulnérabilités sont dues à un défaut de spécification ou de conception (comme par exemple l'oubli d'un cas d'utilisation extrême, l'absence d'interdiction d'écriture dans une zone mémoire critique) ou à une erreur de développement. Ces vulnérabilités sont donc introduites dans le composant avant que celui-ci ne soit disponible ou utilisé. De plus, si un composant contient une vulnérabilité, cette vulnérabilité sera présente sur tous les systèmes comportant ce composant dès la mise en service de ce dernier. Cette caractéristique est très intéressante et permet d'étudier l'évolution de l'état des vulnérabilités dans le temps.

Les vulnérabilités liées à la phase opérationnelle sont le résultat d'une mauvaise utilisation du système, ou d'une utilisation ne correspondant pas aux spécifications de celui-ci. Prenons l'exemple simple de la vulnérabilité utilisée dans l'étude des observations d'attaques à travers des pots de miel dans le cadre du projet CADHo [ALA 06a]. La vulnérabilité choisie était celle du mot de passe faible [ALA 06b] : les systèmes d'exploitation permettent de protéger les données du système via une authentification par un couple (nom d'utilisateur – mot de passe). Celui-ci est une protection efficace pour garantir la confidentialité des données du système s'il ne peut être deviné, ou encore facilement retrouvé dans un dictionnaire utilisé pour les attaques de force brute comme cela est étudié dans [ALA 07]. Ces vulnérabilités sont donc intemporelles et ne sont corrigeables que par les utilisateurs eux-mêmes. Leur interaction avec l'environnement du système est très différente des vulnérabilités créées en phase de développement : les repères temporels sont beaucoup moins distincts tandis que les responsabilités de l'utilisateur et de l'administrateur sont, elles, capitales.

Malgré ces divergences, ces deux classes de vulnérabilités ont un point commun : leur origine humaine. Pour approfondir notre analyse, nous nous intéressons donc aux axes de classification des fautes humaines qui sont évoqués dans [AVI 04]. La figure 2.1 est une représentation en arbre de ces axes et est issue de [AVI 04]. Cette représentation permet de mettre en évidence les différentes classes de fautes existantes en considérant les axes de classification des fautes que nous avons détaillés dans le chapitre précédent.

En suivant cette classification, nous allons en premier lieu nous interroger sur les fautes humaines malveillantes, illustrées par la branche de droite sur la figure 2.1. Ces fautes d'origine malveillante ont été créées avec une intention délibérée. La personne malveillante à l'origine de la vulnérabilité va donc connaître son existence avant même que le composant vulnérable ne soit mis en service et utilisé, comme par exemple dans le cas d'une attaque utilisant une cheval de Troie. Ce scénario est donc celui d'une personne malveillante préparant une attaque précise qui pourra être exécutée dès l'inclusion du composant vulnérable dans un système d'information. Cet aspect est donc non négligeable et nous devrons en tenir compte dans notre approche.

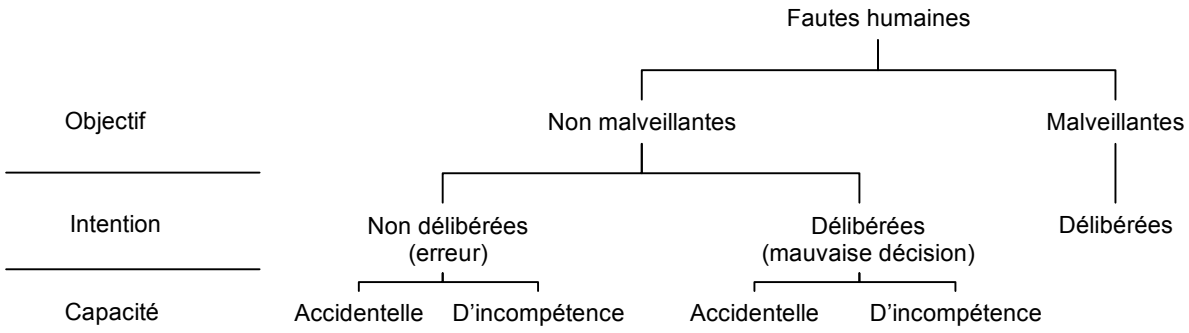


Figure 2.1 : Représentation des fautes humaines en arbre

Les vulnérabilités d'origine non malveillante doivent être étudiées selon l'intention de la faute : la vulnérabilité a-t-elle été créée délibérément ou de façon accidentelle ? Quel impact cela peut-il avoir sur le système ? Que la faute soit due à une erreur ou une mauvaise décision du concepteur ou du développeur, la personne à l'origine de la vulnérabilité n'est pas forcément consciente de l'avoir créée. En conséquence, on peut en déduire que cette personne ne va pas avoir à cœur de l'exploiter ; son objectif n'est pas d'attaquer un système possédant le composant vulnérable. On peut conclure que l'intention et la capacité n'ont pas vraiment d'influence dans le cas d'une vulnérabilité créée sans objectif malveillant. Ce qui se dégage de cette analyse est donc l'importance de considérer l'objectif malveillant ou non malveillant des vulnérabilités.

1.2 Cadre choisi

La notion de vulnérabilité étant une notion regroupant plusieurs types de fautes, nous avons vu qu'il paraît difficile de considérer l'ensemble des vulnérabilités. Adopter une approche globale nous apparaît comme impossible : l'analyse menée dans le paragraphe précédent a mis en lumière les différences d'interaction des vulnérabilités avec l'environnement selon l'axe de classification considéré.

Nous avons vu les grandes différences de contexte entre vulnérabilités créées en phase de développement et vulnérabilités créées en phase d'opération. Il apparaît complexe de vouloir considérer ces deux types de vulnérabilités simultanément en une seule approche car une des contraintes que nous nous sommes imposées dans notre étude et de fournir un processus de mesure qui soit le plus générique possible en restant relativement simple. Ce compromis nous semble pour l'instant difficile dans le cas des vulnérabilités créées en phase opérationnelle. En effet, il ne nous semble pas facilement envisageable de dégager des scénarios typiques et généraux pour cette classe de vulnérabilités. De plus, nous avons évoqué dans le chapitre précédent notre volonté de fournir des mesures utilisant l'unité de temps. Les phases du cycle de vie des vulnérabilités créées durant la phase de développement, depuis leur création jusqu'à l'apparition du correctif ou leur exploitation par la population des attaquants, nous apparaissent comme une base de départ intéressante.

Nous nous plaçons donc dans le contexte de ces vulnérabilités créées durant la phase de développement du composant. Nous choisissons de nous intéresser à une instance de vulnérabilité à la fois.

Dans les prochains paragraphes, nous nous focalisons sur la prise en compte de l'environnement du système en étudiant les facteurs environnementaux qui peuvent avoir une grande influence sur le scénario d'exploitation d'une vulnérabilité. Dans le premier de ces paragraphes, nous faisons l'examen des repères temporels vis-à-vis de la vulnérabilité dont nous avons vu l'importance précédemment : les étapes du cycle de vie de la vulnérabilité.

2 Le cycle de vie de la vulnérabilité

Ce paragraphe présente l'environnement d'un système comportant une vulnérabilité. Nous y détaillons les trois grands axes que nous avons identifiés - le cycle de vie de la vulnérabilité, le comportement de la population d'attaquants et le comportement de l'administrateur du système – et nous étudions les dépendances entre ces éléments.

Dans ce cadre, nous considérons uniquement les vulnérabilités issues de la phase de conception et implémentation du système : ces vulnérabilités sont présentes dans le système dès que celui-ci est disponible à l'utilisation.

2.1 Définitions et études existantes

Nous définissons le cycle de vie comme l'ensemble des événements qui sont à l'origine d'un changement d'état de la vulnérabilité au cours du temps. Plusieurs travaux offrent leurs visions du cycle de vie d'une vulnérabilité, que nous avons résumées dans le tableau 1. Les événements sont classés selon la chronologie adoptée par les auteurs. Les événements indiqués en gras indiquent qu'ils sont considérés par plusieurs auteurs à des instants chronologiques différents. En 2000, dans [ARB 00], les auteurs font partie des premiers à évoquer le cycle de vie de la vulnérabilité en le définissant par une liste d'événements. Les auteurs incluent la naissance de la vulnérabilité, sa découverte (en spécifiant que celle-ci est confondue avec la naissance dans le cas d'une faute intentionnelle et considérant ainsi les deux branches de l'arbre de la figure 2.1), sa publication, sa correction, sa connaissance du grand public, l'existence d'un kit d'exploitation et enfin sa mort. Dans [FIS 03], l'auteur aborde le cycle de vie d'une manière très différente en prenant en considération beaucoup plus d'événements. Il évoque l'événement de redécouverte de la vulnérabilité ou fuite d'information profitant à une personne malveillante. Il découpe également le processus de mise au point d'un kit d'exploitation en trois événements distincts : l'existence du concept de l'attaque, l'existence d'une attaque aboutissant à un succès, et l'automatisation de l'attaque. Il ajoute également deux événements liés au correctif en plus de sa publication : son application et la publication d'un correctif "complet". Dans [RES 05], l'auteur présente une version plus épurée du cycle de vie de la vulnérabilité, se rapprochant de celle de [ARB 00], en n'y listant que la naissance, la découverte, la publication, l'exploitation et la correction de la vulnérabilité. Il est à noter que l'auteur place néanmoins la mise au point et l'utilisation du kit d'exploitation après la publication de la vulnérabilité et non pas après la publication du correctif. Dans [FRE 06], l'auteur se limite encore plus en ne considérant que les événements de découverte, exploitation, publication et publication du correctif. Là encore, il est important de remarquer que la phase d'exploitation de la vulnérabilité est cette fois-ci considérée juste après la découverte de la vulnérabilité. Cependant, dans [FRE 09], l'auteur ajoute l'événement d'installation du patch à sa défintion du cycle de vie de la vulnérabilité. Il précise également que l'ordre des événements varie en fonction de la vulnérabilité

considérée. Enfin, dans [JON 07], l'auteur reprend la notion de connaissance de la vulnérabilité par le grand public, rencontrée dans [ARB 00], mais la place avant la publication du correctif.

Arbaugh, Fithen, McHugh (2000)	Naissance	Découverte			Publication			
Fischbach (2003)		Découverte de la faille	Redécouverte de la faille ou fuite		Publication		Preuve	Création du kit d'exploitation
Rescorla (2005)	Naissance	Découverte			Publication			Création du kit d'exploitation
Frei (2006)		Découverte		Création du kit d'exploitation	Publication			
Jones (2007)	Naissance	Découverte			Publication	Publicité		

Arbaugh, Fithen, McHugh (2000)		Correction	Publication	Création du kit d'exploitation			Mort
Fischbach (2003)	Création du kit d'exploitation	Correctif disponible			Correctif appliqué	Correctif complet	
Rescorla (2005)		Publication du correctif					
Frei (2006)		Correctif					
Jones (2007)	Création du kit d'exploitation	Correction					Mort

Tableau 2.1 : Etapes du cycle de vie d'une vulnérabilité ordonnées dans le temps

Par ailleurs, dans [DUM 98], les auteurs adoptent une approche totalement différente en ce qui concerne la définition du cycle de vie de la vulnérabilité en le décomposant en trois notions liées par un lien de cause à effet : cause, impact et réparation. Ce cycle de vie ne tient pas compte des mêmes notions car il ne se rattache pas à des événements temporels précis. Il ne convient donc pas à ce que nous voulons caractériser par notre approche.

2.2 Les événements du cycle de vie considérés

Parmi les événements cités par les définitions rencontrées, quels sont ceux qui composent le cycle de vie de la vulnérabilité tel que nous allons l'utiliser ? Notre but est de produire un modèle paramétrable qui soit aussi générique que possible tout en ne dépassant pas un certain degré de complexité.

Une de nos exigences est donc que les étapes du cycle de vie de la vulnérabilité que nous considérons doivent être ordonnées dans le temps pour maintenir des repères temporels aisément utilisables. Cet ordre doit être valide quelque soit la vulnérabilité considérée pour s'adapter au désir de généricité de l'approche. De ces études sur le cycle de vie d'une vulnérabilité, nous pouvons tirer la conclusion que quatre événements sont capitaux pour caractériser le cycle de vie d'une vulnérabilité. Ce sont :

- la découverte de la vulnérabilité ;
- la publication de la vulnérabilité ;
- la publication du correctif de la vulnérabilité ;
- l'apparition du kit d'exploitation.

Nous avons pu remarquer l'importance de la phase d'exploitation de la vulnérabilité, présente dans chacun des travaux que nous avons cités. Cependant, nous avons pu remarquer également que cette phase d'exploitation était considérée à plusieurs moments différents dans les approches décrites. Il apparaît donc que cette phase d'exploitation de la vulnérabilité ne peut pas être datée et ordonnée dans le temps relativement aux autres événements. En effet, la phase d'exploitation peut très bien intervenir dès la vulnérabilité découverte, ou, à l'extrême inverse, bien après la publication du correctif. De plus, il nous apparaît nécessaire d'étudier l'impact de cet événement sur la population des attaquants. Nous choisissons donc de décrire cet événement du cycle de vie de la vulnérabilité dans la section consacrée à la population des attaquants (cf partie 3).

Les trois moments de la vie d'une vulnérabilité que sont la découverte, la publication et la publication du correctif sont des instants importants de par l'impact qu'ils ont sur l'état de connaissance de l'existence la vulnérabilité, et comment cette connaissance est susceptible de se propager. Les termes appropriés à ces événements peuvent être soumis à interprétation, c'est pourquoi nous précisons leur sens par les définitions suivantes.

2.2.1 La découverte de la vulnérabilité

Nous définissons la découverte de la vulnérabilité comme l'instant où la vulnérabilité est découverte pour la première fois sachant que 1) la vulnérabilité existe et 2) le composant vulnérable est disponible à l'utilisation. Cette découverte peut être faite par une personne malveillante ou non malveillante. Cet événement traduit l'instant à partir duquel au moins une personne connaît l'existence de la vulnérabilité. Dans [FRE 09], cet événement est défini comme le premier instant où la vulnérabilité est considéré comme étant la source d'un risque potentiel pour la sécurité des systèmes d'information. Cette définition est en accord avec la nôtre.

2.2.2 La publication de la vulnérabilité

Nous avons vu que l'événement de publication de la vulnérabilité semble être un événement central du cycle de vie de la vulnérabilité. Preuve en est que chacun des cycles de vie que nous avons étudiés évoque cet événement quand d'autres ne sont évoqués que par une faible proportion des auteurs (cf. tableau 2.1). De plus, cet événement signifie qu'il est possible à tout un chacun de connaître la vulnérabilité, que l'on soit malveillant ou non. L'occurrence de cet événement possède donc l'avantage majeur de permettre aux utilisateurs et administrateurs de système de se montrer plus vigilants lorsqu'ils ont connaissance de l'existence de la vulnérabilité. La publication peut être considérée comme une forme de pression pour une publication rapide du correctif par le producteur du composant vulnérable [FRE 09].

Cependant, l'événement de publication de la vulnérabilité a aussi la conséquence néfaste de permettre aux attaquants d'avoir connaissance de la vulnérabilité et de mettre à profit cette connaissance pour perpétrer des attaques en l'exploitant.

Ces deux conséquences donnent une dimension particulière au phénomène de publication de la vulnérabilité. Il arrive ainsi que les producteurs du composant vulnérable cherchent à retarder le plus possible la publication de la vulnérabilité afin d'écourter le plus possible le laps de temps entre la publication de la vulnérabilité et la mise à disposition du correctif.

Par ces conséquences importantes, l'événement de publication de la vulnérabilité représente donc un événement déterminant dans le cycle de vie de la vulnérabilité. D'après [FRE 09], il peut se définir comme le premier instant où il existe une information validée à propos de la vulnérabilité qui soit accessible au public. Nous adhérons à cette définition et spécifions les caractéristiques suivantes :

- l'information sur la vulnérabilité est disponible librement ;
- l'information sur la vulnérabilité est publiées par une source reconnue large ;
- la vulnérabilité a fait l'objet d'une analyse par des experts.

Cette définition se distingue de celle énoncée dans [FRE 09] car nous n'exigeons pas que la source soit indépendante et l'analyse des experts n'a pas nécessairement besoin d'avoir fait l'objet d'une évaluatin du niveau de risques de la vulnérabilité. En effet, les sites des producteurs des composants vulnérables semblent également être une source correcte pour la publication de vulnérabilités. Par exemple une société comme Microsoft diffuse de façon régulière (le premier mardi du mois) les correctifs liés aux vulnérabilités de son système d'exploitation et des logiciels qu'elle propose.

Il représente donc l'instant de publication officielle de la vulnérabilité par le producteur du composant vulnérable ou par un centre d'alerte et de recensement de vulnérabilités comme CVE [CVE].

2.2.3 La publication du correctif de la vulnérabilité

Enfin, l'événement de publication du correctif de la vulnérabilité a lieu lorsque ce dernier 1) existe et 2) est disponible et accessible par tous les utilisateurs via les sources considérées pour la publication de la vulnérabilité. L'événement de publication du correctif peut avoir lieu de manière simultanée avec l'événement de publication de la vulnérabilité. Dès lors que le correctif est disponible, l'administrateur a la possibilité de supprimer ou masquer la vulnérabilité du composant vulnérable de son système.

Ces trois événements sont donc incontournables. Etudions maintenant les autres événements évoqués par certaines des approches existantes.

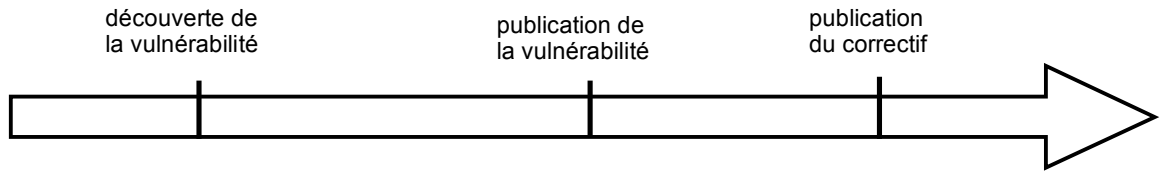


Figure 2.2 : Les événements du cycle de vie de la vulnérabilité

La naissance de la vulnérabilité est également un élément important, évoqué par la majorité des approches. Cependant, plus que l'événement de naissance de la vulnérabilité, il apparaît comme important le fait qu'elle existe.

Certains événements sont abordés par les définitions de cycle de vie de la vulnérabilité que nous avons évoquées, tels que la notion de connaissance d'une vulnérabilité par le grand public. Il est difficile de quantifier cet événement et de savoir quand il a lieu : quelle proportion de la population faut-il considérer pour pouvoir considérer que la vulnérabilité est connue du grand public ? Il apparaît difficile de quantifier cette notion dans le temps. De plus, une vulnérabilité est connue du grand public

lorsque le phénomène d'attaque qui lui est lié devient important. Nous ne considérons donc pas cette phase comme pertinente dans le cadre que nous considérons.

Il en est de même pour les événements associés au correctif décrits dans l'approche de [FIS 03], ainsi que pour le phénomène de mort de la vulnérabilité [ARB 00, JON 07]. De plus cet événement de mort de la vulnérabilité ne nous paraît pas pertinent pour notre approche : en effet, comment affirmer qu'une vulnérabilité n'existe plus si on considère plus qu'un système unique ? Cela signifie qu'il n'existe plus aucun composant affecté par cette vulnérabilité. Il semble difficile de pouvoir affirmer l'occurrence d'un tel événement autant que d'en définir l'instant d'occurrence.

Enfin, le phénomène de redécouverte de la vulnérabilité décrit dans [FIS 03] n'est pas une étape obligatoire de cycle de vie de la vulnérabilité. Pour cette raison, nous n'en tenons pas compte.

Notre cycle de vie de la vulnérabilité comporte donc trois événements distincts et ordonnés dans le temps, comme illustré sur la figure 2.2.

3 La population des attaquants

3.1 Caractérisation et influence sur le système

La population des attaquants est le deuxième facteur environnemental que nous considérons. Cependant, qui sont ces attaquants et comment pouvons-nous caractériser leurs actions et leur comportement ? La population des attaquants n'est pas homogène. Dans [CER 05, ROG 06], les auteurs proposent une classification à deux dimensions des attaquants en tenant compte de leur motivation et de leur degré de compétence. Les motivations d'un attaquant à vouloir exploiter une vulnérabilité peuvent être très diverses. Plusieurs motivations possibles sont évoquées dans les profils définis dans [CER 05] :

- Le jeu ou le défi : la recherche de performance ;
- La vengeance ;
- Le vol à des fins spéculatives : on peut considérer soit le vol de fonds financiers, soit le vol d'informations confidentielles à des fins de vente à la concurrence ou de chantage.

Ces différentes motivations sont incluses dans les sept classes d'attaquants définies, mais peuvent être considérées uniquement sur une échelle nominale. De plus, l'auteur ne fournit pas la proportion d'attaquants associée à chaque classe. Cela ne permet pas de prendre en considération une ou plusieurs classes majoritaires dans la population des attaquants, ni de considérer une des motivations évoquées comme une caractéristique exploitable et évaluable quantitativement.

Les approches décrites dans [DAC 96, MCQ 06] se concentrent davantage sur les compétences de l'attaquant en considérant un degré qualitatif de compétence ou en le reliant au nombre de vulnérabilités que celui-ci sait exploiter. Dans ces deux approches, chaque classe d'attaquant est considérée séparément : la proportion de chaque classe de populations d'attaquants n'a donc pas besoin d'être évaluée.

Enfin, dans [ALA 07], l'auteur ne considère que deux catégories d'attaquants : les *script kiddies* et les *black hats*. Les premiers sont décrits comme des attaquants novices, utilisant des kits d'exploitation disponibles dans la communauté. Les seconds sont des experts qui mettent au point les outils utilisés par les premiers.

Nous appelons "kit d'exploitation" ce qui permet à un attaquant d'exploiter une vulnérabilité sans avoir "besoin de comprendre" ce qu'il fait. Nous citons par exemple les méthodes dites "recettes de cuisine" qui expliquent pas à pas les commandes à taper par l'attaquant, les scripts prêts à exécuter, les outils, etc. Cette notion correspond à celle évoquée par le terme anglais *exploit*.

D'après les observations obtenues par l'étude décrite dans [ALA 07], il apparaît que les attaquants novices (*script kiddies*) sont considérablement plus nombreux que les experts (*black hats*). Ces attaquants novices ont besoin de posséder le kit d'exploitation pour pouvoir exploiter la vulnérabilité associée. L'existence de ce kit d'exploitation va donc avoir une influence importante sur le nombre d'attaquants susceptibles d'exploiter la vulnérabilité. Cette observation est validée par les données fournies par [RAM 07]. Dans cet article, les auteurs comparent les données récoltées par le projet Leurré.com [LEU] aux dates de parution du kit d'exploitation sous la forme des plugins d'attaque de l'outil *metasploit*[12] [MET]. Ils constatent une augmentation très importante des attaques exploitant une certaine vulnérabilité aux instants précédant et suivant l'apparition du plugin contenant le script d'attaque de l'outil *metasploit*. Cette étude met donc en évidence l'importance du phénomène d'attaque utilisant un kit d'exploitation. Mais elle met également en évidence le nombre très faible d'attaques avant la parution du kit d'exploitation.

Ce résultat illustre l'utilisation importante de ces kits d'exploitation sans lesquels de nombreux attaquants n'auraient sans doute pas les compétences nécessaires pour exploiter la vulnérabilités. Cette conclusion semble cohérente avec les résultats de l'étude détaillée dans [ALA 07] qui met en avant la très forte proportion d'attaquants novices par rapport aux attaquants experts.

Ces études mettent donc en avant l'importance de l'événement de mise au point d'un kit d'exploitation par la population d'attaquants. Cette parution du kit d'exploitation permet aux nombreux attaquants novices d'exploiter la vulnérabilité sans parfois même avoir à comprendre l'attaque elle-même. Il est donc capital de prendre cet événement en considération car il implique une augmentation très significative du nombre d'attaques, jusqu'alors là issues d'attaquants experts et qui peut être en comparaison négligeable.

Pouvoir caractériser plus en détail le comportement de la population d'attaquants permettrait de considérer d'autres paramètres permettant ainsi d'affiner les mesures quantitatives que nous tentons de mettre au point. Cependant, les données restent peu nombreuses, dues à la difficulté d'observer cette population.

[12] *metasploit* est une plateforme de test de présence de vulnérabilités et de tentative d'intrusion du système. Les scripts d'attaque sont écrits sous forme de plugin pour permettre une mise à jour constante de l'outil.

3.2 L'influence de l'existence de la preuve de concept

Nous nous sommes basés, juqu'à présent, sur le fait que le kit d'exploitation est mis au point par la population des attaquants. Cependant, un événement apparaissant dans le cycle de vie de la vulnérabilité décrit dans [FIS 03] met en évidence l'influence de l'existence de la preuve de concept. Cet événement se produit lorsque la personne qui découvre la vulnérabilité sans intention malveillante, montre que la vulnérabilité peut être exploitée en produisant le code permettant son exploitation.

L'existence de cette preuve de concept permet souvent à la population des attaquants de la détourner à des fins malveillantes et de s'en servir pour perpétrer des attaques. En ne considérant pas les conséquences de cet événement, notre approche ne nous permet de considérer que les vulnérabilités pour lesquelles le kit d'exploitation est 1) mis au point par la population d'attaquants avant ou après l'événement de publication et 2) une réutilisation malveillante de la preuve de concept avec l'hypothèse que celle-ci est diffusée uniquement lors de la publication de la vulnérabilité.

En effet, il paraît peu probable que cette preuve de concept permettant l'exploitation de la vulnérabilité puisse apparaître avant la publication de la vulnérabilité. En considérant l'hypothèse inverse, la personne qui permet la publication de la preuve de concept n'est pas malveillante mais va permettre que la preuve de concept soit détournée de façon malveillante. C'est une nuance qui n'est pas prise en considération dans nos scénarios. Cependant, cette personne non malveillante mais imprudente ou inconsciente, qui permet la diffusion de la preuve de concept ne peut-elle pas être assimilée comme personne malveillante et correspondre ainsi au scénario de découverte malveillante ?

L'existence de la preuve de concept ne contredit en rien la logique des scénarios que nous avons décrits.

3.3 Interdépendance avec le cycle de vie – Scénarios de fonctionnement

Dans ce paragraphe, nous allons nous interroger quant aux dépendances entre les deux phénomènes caractérisés précédemment : le cycle de vie de la vulnérabilité et le comportement de la population d'attaquants. Cette réflexion se base sur l'observation que nous avons faite dans l'étude du cycle de vie de la vulnérabilité : toutes les définitions du cycle de vie que nous avons rencontrées évoquaient la phase d'exploitation de la vulnérabilité. Cependant, aucune étude ne plaçait cet événement au même endroit d'un point de vue chronologique, ce qui a motivé notre volonté d'isoler ce phénomène. Nous avons caractérisé cette phase relevant du comportement des attaquants par l'événement de création du kit d'exploitation. Néanmoins, il semble important de rechercher les relations de dépendance entre ces deux caractéristiques du phénomène d'attaque.

Le premier événement auquel nous nous intéressons est le phénomène de découverte de la vulnérabilité. Comme nous l'avons précisé dans notre définition, cette découverte peut être faite autant par une personne malveillante qu'une personne non malveillante. L'origine de la découverte a donc une influence sur l'événement de mise au point du kit d'exploitation. Cet événement va donc être à l'origine de deux scénarios distincts qu'il nous faut prendre en compte. Ces scénarios sont détaillés par les figures 2.3 et 2.4. Pour chacun de ces schémas, les événements chronologiques du cycle de vie de la vulnérabilité et l'événement de création du kit d'exploitation sont ordonnés dans le temps. Les intervalles de temps entre les événements ne sont pas quantifiés : le laps de temps entre deux événements peut être très court (inférieur à un jour). Les flèches entre les axes indiquent le lien de causalité entre l'événement d'origine et l'événement désigné.

L'évolution des états de connaissance de la vulnérabilité pour les populations malveillante (symbolisée par le personnage rouge) et non malveillante (symbolisée par le personnage bleu) est également indiqué.

Plaçons-nous dans la première hypothèse possible : l'origine de la découverte est non malveillante. La population des attaquants ne peut rien savoir de l'existence de la vulnérabilité. Seul l'événement de publication de la vulnérabilité permettra à la population des attaquants de prendre connaissance de l'existence de la vulnérabilité. C'est à partir de cet instant que ces derniers seront à même de mettre au point un kit d'exploitation ou de détourner de manière malveillante la preuve de concept. Ce scénario est illustré dans la figure 2.3.

Considérons maintenant la seconde hypothèse : la découverte est faite par une personne malveillante. A partir de cet instant, la proportion de la population des attaquants qui va avoir connaissance de la vulnérabilité va peu à peu augmenter. Dès qu'un attaquant aura été suffisamment compétent pour mettre au point un kit d'exploitation, ce dernier va être utilisé par les attaquants connaissant la vulnérabilité. Ce sont ces attaques, de plus en plus nombreuses, utilisant ce kit d'exploitation qui informeront la population non malveillante de l'existence de la vulnérabilité. Ceci provoquera le phénomène de publication de la vulnérabilité. Ce scénario est illustré par la figure 2.4. Ce scénario prend donc en compte les vulnérabilités de type *0-day*.

La distinction de ces deux scénarios permet d'inclure un aspect important que nous avons souligné au début de ce chapitre : l'intention de la personne à l'origine de la vulnérabilité. Si la vulnérabilité a été créée intentionnellement par une personne malveillante, celle-ci est au courant de la présence de la vulnérabilité avant même que le composant vulnérable ne soit disponible à la vente ou au téléchargement. Néanmoins, cette connaissance lui est inutile tant que le composant n'est pas utilisé, donc susceptible d'être exploité. On peut donc inclure ce cas de figure dans le scénario de découverte par une personne malveillante, qui découvrirait la vulnérabilité dès la disponibilité du composant vulnérable et qui mettrait au point un kit d'exploitation immédiatement.

Ces deux scénarios mettent en évidence l'interdépendance forte entre les deux facteurs environnementaux que nous avons présentés. Ces facteurs sont des facteurs globaux qui sont communs à tous les systèmes. Cependant, on peut s'interroger sur l'existence de facteurs environnementaux locaux : quel paramètre peut faire que deux systèmes identiques n'encourent pas les mêmes risques vis-à-vis de la sécurité ? Dans le paragraphe suivant, nous présentons le dernier facteur environnemental que nous considérons qui est l'administrateur du système.

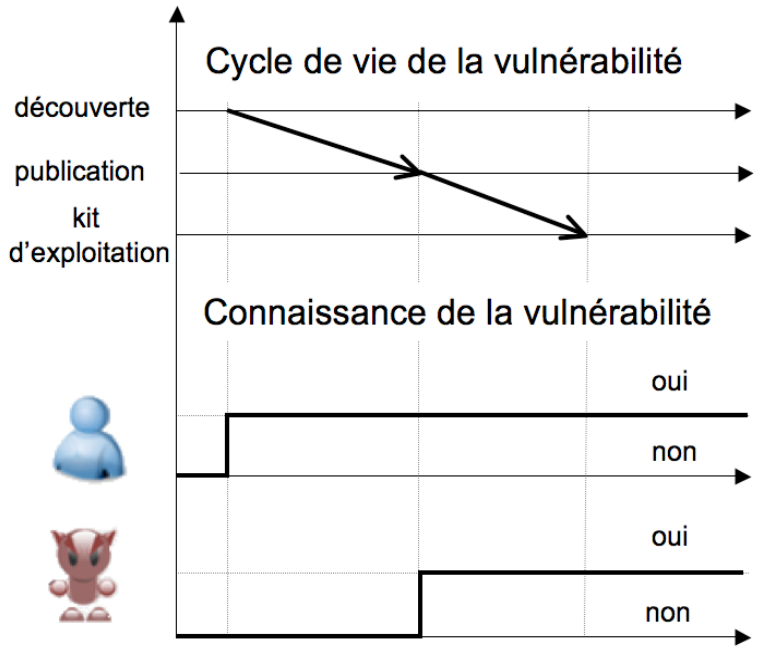


Figure 2.3 : Scénario de découverte par une personne non malveillante

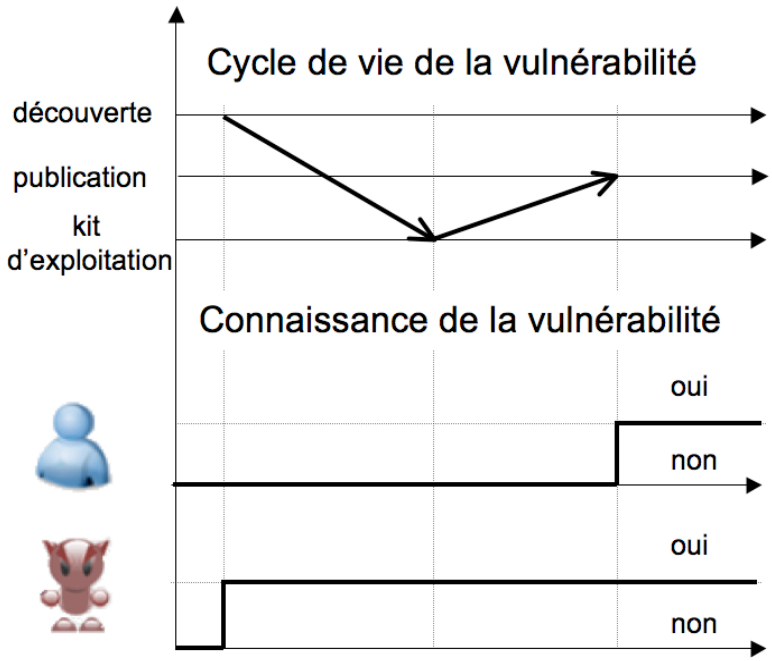


Figure 2.4 : Scénario de découverte par une personne malveillante

4 Le comportement de l'administrateur

4.1 Intérêt

L'administrateur d'un système d'information a pour but de veiller à la bonne application de la politique de sécurité du système. Dans le cadre d'une entreprise, l'administrateur se doit d'avoir une bonne connaissance des dangers liés à la sécurité. Cependant, un administrateur est un facteur environnemental et humain : il peut donc être faillible. Plus encore, dans le cas des systèmes d'information domestiques, l'administrateur du système est l'utilisateur du système lui-même. Dans ces circonstances, celui-ci n'a pas forcement connaissance des dangers liés à la sécurité du système et n'est pas nécessairement formé pour y faire face et protéger efficacement son système. Que ce soit par incompétence ou par laxisme, un administrateur peut maintenir son système en danger face aux vulnérabilités qu'il peut contenir.

Bien sûr, un administrateur peut s'aider d'outils de gestion de la sécurité : antivirus et pare-feu peuvent aider un administrateur à maintenir son système non vulnérable et à empêcher certaines attaques. De plus, ces outils peuvent être automatisés afin de récupérer au plus tôt les correctifs de sécurité disponibles en effectuant des mises à jour périodiques. Nous pouvons faire l'hypothèse qu'un administrateur rigoureux choisira des outils de gestion de la sécurité et les configurera afin d'examiner chaque comportement suspect et de récupérer les mises à jour fréquemment. A l'inverse, un administrateur laxiste n'installera pas ces outils ou ne les configurera pas, par inconscience ou méconnaissance des menaces liées à la présence de vulnérabilités sur son système.

Prenons l'exemple d'une personne non sensibilisée aux problèmes de sécurité qui acquiert un nouvel ordinateur personnel tel qu'ils sont disponibles dans le commerce. Cette personne est donc l'administrateur de son système. Sur son système sont installés par défaut un système d'exploitation, plusieurs logiciels et un antivirus commercial en version de démonstration. N'étant pas sensibilisé aux dangers liés à la sécurité, l'administrateur de ce système ne configure pas le pare-feu du système, et n'automatise pas la recherche des mises à jour du système d'exploitation. Au bout de la période d'essai de l'antivirus, l'administrateur ne ressent pas le besoin de dépenser de l'argent pour le renouveler et n'a pas les connaissances pour trouver et installer un antivirus gratuit. Son système ne sera donc plus protégé contre les vulnérabilités dont le correctif sera publié après la date d'expiration de la version d'essai de son antivirus. Ce scénario simple montre à quel point une personne inconsciente des risques liés à la sécurité des systèmes peut avoir une influence sur les risques encourus par le système.

Bien entendu, dans le cas d'un système multi-utilisateurs, l'administrateur n'est pas l'unique source d'interaction locale avec le système. Il n'est donc pas le seul facteur environnemental agissant sur la sécurité du système. Les utilisateurs représentent eux aussi un facteur environnemental très important et leur comportement peut être à la source de grandes défaillances de sécurité. Par exemple, il n'est pas rare qu'un utilisateur choisisse un mot de passe trop faible ou cache celui-ci sous son clavier. Des études se sont penchées l'influence de l'utilisateur sur le système. Néanmoins, il est important de préciser que l'utilisateur est un facteur environnemental important lorsque l'on considère les vulnérabilités de configuration, que nous ne prenons pas en compte dans notre approche.

4.2 Choix comme paramètre d'étude

Se focaliser sur l'administrateur du système semble avoir plusieurs intérêts. En premier lieu : ce facteur environnemental est celui qui est spécifique à chaque système. Il nous apparaît donc important de l'étudier plus particulièrement. En second lieu, ce facteur environnemental tient un rôle important dans un de nos objectifs. En effet, nous avons vu que nous pouvions envisager deux points de vue différents pour évaluer la sécurité d'un système d'information. Ces deux types de mesures répondent à deux questions distinctes. Une mesure déterminant une mesure de distance à un objectif répond à une question telle que : "quel effort ai-je à fournir, ou combien de temps dois-je consacrer pour atteindre et maintenir le niveau de sécurité que je souhaite ?" Prenons un exemple simple : une entreprise soucieuse de la sécurité de son système d'information, veut maintenir un niveau de sécurité choisi. L'administrateur doit être suffisamment compétent et surtout disponible pour être capable d'assurer le niveau de sécurité voulu.

Une telle évaluation permet à un système d'information d'avoir un niveau de sécurité maintenu tout en optimisant le coût pour l'administrateur et l'entreprise propriétaire du système. De plus, ce choix permet d'ouvrir notre approche vers les perspectives envisagées, à savoir la prise en compte des vulnérabilités non considérées. En effet, nous avons vu dans la première partie de ce chapitre que ces vulnérabilités résultant de la phase opérationnelle avaient une plus grande interaction avec l'administrateur du système, directement impliqué dans la configuration de celui-ci. Approfondir cet aspect nous permet donc d'étudier par la même occasion l'extension de notre approche à ces vulnérabilités.

4.3 Actions de l'administrateur

Dans cette partie, nous présentons les différentes actions possibles de l'administrateur afin de pouvoir déterminer celles à prendre en considération dans notre approche. Les actions qui nous intéressent doivent toutes avoir un impact significatif sur la présence de la vulnérabilité sur le système.

La première action à laquelle nous nous intéressons est l'installation du composant vulnérable lui-même. En effet, la vulnérabilité peut exister, être exploitée de façon intensive ou encore être corrigée via un correctif publié par le producteur du composant vulnérable, toutes ces circonstances n'auront aucun impact si le composant vulnérable n'a pas été installé sur le système. Dans l'optique d'une étude focalisée sur une vulnérabilité bien identifiée et particulière, cette action et les deux possibilités qui en découlent (le composant est installé ou le composant n'est pas installé sur le système) doivent être considérées séparément. Si la vulnérabilité est contenue dans un composant de base, il revient à examiner la compatibilité du système avec la vulnérabilité. Il est tout de même important de noter que dans ce cadre, l'action de l'administrateur n'est plus le seul paramètre influençant cette action, le choix de la configuration de base du système est lui aussi un facteur prépondérant.

La seconde catégorie d'actions possibles pour l'administrateur du système est l'application du correctif de la vulnérabilité. Cette action peut avoir plusieurs motivations et peut se faire à plusieurs moments de la vie opérationnelle du système. Ce sont ces circonstances que nous détaillons dans ce paragraphe. L'application du correctif de la vulnérabilité peut se faire lors d'une mise à jour : l'administrateur cherche à appliquer tous les correctifs disponibles à son système sans se focaliser particulièrement sur une vulnérabilité particulière. Cette action a lieu régulièrement et est motivée par un souci de

sécurité et non par l'urgence de correction face à une compromission du système. Dans ce second cas de figure, l'administrateur peut chercher à appliquer un correctif précis correspondant à la vulnérabilité qui a été exploitée. Cette action n'est pas une action routinière mais bien ciblée de la part de l'administrateur. On peut supposer que ce scénario a lieu essentiellement dans le cas d'une compromission du système par cette vulnérabilité. Ces actions peuvent intervenir n'importe quand dans la vie du système, à partir de l'instant où le correctif est disponible. Ces actions sont caractéristiques de la rigueur de l'administrateur.

Une catégorie d'actions de l'administrateur est la restauration et la réparation des données. Ces actions interviennent lorsque le système a été victime d'une attaque. Ces actions ne sont pas anodines dans le sens où 1) les données ne sont pas toujours récupérables ; 2) restaurer les données à un coût en temps pour l'administrateur du système.

A partir de ces actions, nous allons nous interroger sur la quantification de ce comportement de l'administrateur. Dans le paragraphe suivant, nous présentons les études existantes et cherchons les axes de quantification.

4.4 Etudes existantes et quantification

Cependant, comment pouvons-nous quantifier ce comportement ? Il existe très peu d'études se penchant sur cette problématique et ayant cherché à produire des indices quantitatifs de l'influence du comportement humain sur la sécurité du système.

Quelques études se sont penchées sur le comportement de l'utilisateur. Dans le cas d'un système d'information privé et personnel, ces études considèrent que les rôles d'utilisateur et d'administrateur sont confondus. Certaines de ces études tentent de rapprocher le comportement vis-à-vis de la sécurité avec les comportements humains vis-à-vis de la santé, comme par exemple [NGK 08]. En effet, le parallèle peut être fait entre ces deux comportements : prenons l'exemple d'une personne mettant à jour régulièrement son système, celle-ci peut être assimilée à une personne prenant soin de maintenir son carnet de vaccination parfaitement à jour. Dans le cadre des vulnérabilités de configuration ou de mode d'utilisation, il en est de même : une personne adoptant un mot de passe fort peut être comparée dans le domaine de la santé à une personne adepte de médecine préventive. Dans cette approche, des statistiques sont basées sur un questionnaire distribué à plus d'une centaine de personnes et abordant des thèmes subjectifs et objectifs. Les auteurs étudient plusieurs aspects et concluent sur la pertinence de leur impact sur le comportement lié à la sécurité : une des principales conclusions est que la sécurité est vue comme une contrainte, ce qui incite les utilisateurs au laxisme. On peut cependant noter que cette étude se focalise sur les utilisateurs d'un système dans le cadre professionnel. Une autre étude, détaillée dans [WAS 07], tente de modéliser les conséquences du comportement de l'utilisateur en distinguant les bons et les mauvais utilisateurs. Mais cette approche non plus ne propose pas de mesure quantitative pour le comportement de l'utilisateur ou de l'administrateur.

Nous avons fait le choix de produire des mesures temporelles, nous pouvons quantifier ce comportement en considérant les temps de réaction de ce dernier. A quelle fréquence fait-il les mises à jours de sécurité de son système ? Après combien de temps réagit-il lors d'une attaque ? Ces paramètres quantitatifs sont un ensemble de données intéressant pour modéliser et quantifier le comportement de l'administrateur.

Lié aux autres facteurs que nous avons précédemment décrits, ce comportement de l'administrateur va jouer un rôle important.

4.5 Influences mutuelles entre facteurs environnementaux

Dans la partie précédente, nous avons décrit le comportement de l'administrateur et insisté sur l'action de mise à jour du système. Selon son degré de rigueur, l'administrateur permet une mise à jour du système et de ses outils de protection à une fréquence plus ou moins grande.

Cependant, il est impossible pour l'administrateur d'appliquer un correctif pour une vulnérabilité existante tant que celui-ci n'a pas été publié. Le cycle de vie de la vulnérabilité impose donc des limites à l'action de l'administrateur. Cependant, on peut souligner que cet événement, autant que ceux du cycle de vie, peut influencer l'attention que l'administrateur va accorder à la sécurité du système. Par exemple, un administrateur déjà rigoureux le deviendra d'autant plus s'il prend connaissance de la publication d'une vulnérabilité, ou de l'existence d'une série d'attaques importante via un kit d'exploitation, ou d'un vers se propageant sur Internet.

Les dépendances entre facteurs environnementaux et leurs influences sur le système sont décrites dans la figure 2.5. On retrouve la triple influence du cycle de vie de la vulnérabilité sur l'événement de mise au point du kit d'exploitation, sur le comportement de l'administrateur et sur l'état du système lui-même. La mise au point du kit d'exploitation influe à la fois le cycle de vie de la vulnérabilité et l'état du système. Enfin, on retrouve l'influence de l'administrateur décrite dans cette partie.

Les influences de ces trois facteurs environnementaux sur le système, dans le contexte de la présence d'une vulnérabilité dans le système, sont des éléments importants à considérer dans notre approche. En effet, les répercutions de ces événements externes au système vont avoir des conséquences sur l'état su système vis-à-vis de la sécurité. C'est cet état du système que nous allons évaluer.

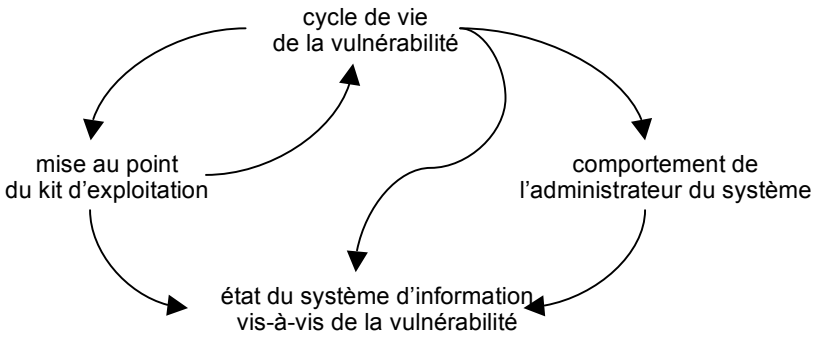


Figure 2.5 : Influences des facteurs environnementaux

5 Mesure de la zone de haut risque : une première mesure simple

Nous avons vu les différentes étapes du cycle de vie : la découverte de la vulnérabilité, la publication de la vulnérabilité et la publication du correctif de la vulnérabilité. Ces événements sont ordonnés dans le temps. Cependant, nous avons vu également que la publication du correctif n'est pas une condition suffisante à la protection du système pour que celui-ci soit protégé. Il faut que l'administrateur du système ait installé le correctif.

Dans [FRE 06], les auteurs associent des estimations qualitatives du danger en fonction du cycle de vie de la vulnérabilité qu'ils y décrivent. Ils estiment que le système est en grand danger entre les événements de découverte et de publication de la vulnérabilité car une grande proportion de la population n'a pas connaissance de l'existence de la vulnérabilité. Cependant, dans [FIS 03], les auteurs montrent qu'une grande proportion des attaques a lieu juste après la publication de la vulnérabilité. Nous sommes enclin à suivre cette interprétation : lorsque la vulnérabilité est connue par une grande proportion de la population des utilisateurs, elle le sera également par une grande proportion de la population d'attaquants. En considérant cette tendance, nous supposons que le système sera en grand danger vis-à-vis d'une vulnérabilité entre l'événement de publication de la vulnérabilité et l'événement d'application du correctif par l'administrateur du système. Nous appelons cet intervalle de temps la zone de haut risque qui se mesure de la manière suivante :

$$T_{HR} = (t_c - t_p) + (t_{app} - t_c) = t_{app} - t_p$$

Les symboles t_p, t_c et t_{app} désignent respectivement les instants de publication de la vulnérabilité, de publication du correctif et d'application du correctif. Un aspect important de cette mesure est qu'elle prend en considération deux facteurs environnementaux : le cycle de vie de la vulnérabilité et le comportement de l'administrateur du système. Cet intervalle de temps est illustré par la figure 2.6.

Cette mesure ne tient pas compte de l'apparition du kit d'exploitation, qui, comme nous l'avons vu dans les parties précédentes, ne peut pas être ordonné dans le temps vis-à-vis des événements du cycle de vie de la vulnérabilité. Cette mesure ne suffit donc pas à une quantification de la sécurité considérant l'ensemble des facteurs environnementaux, mais elle permet une première quantification simple de la durée de risque : c'est-à-dire la durée pendant laquelle le système est vulnérable et où l'existence d'un kit d'exploitation devient très probable.

Pour illustrer cette mesure, nous prenons l'exemple de l'épidémie du vers Slammer. Ce vers exploite la vulnérabilité de Microsoft SQL Server 2000, décrite dans [MIC 02]. Elle permet d'effectuer un débordement de mémoire tampon (*buffer overflow* en anglais). La vulnérabilité et son correctif ont été publiés le même jour : le 24 juillet 2002. L'épidémie liée au vers Slammer a commencé le 25 janvier 2003, soit 185 jours plus tard. Il agit de la manière suivante sur le système : 1) il obtient les droits d'utilisation de trois fonctions de l'API Windows sur le système (*GetTickCount*, *socket* et *sendto*) ; 2) il envoie son code (376 octets) vers des adresses IP aléatoires – il parvient à envoyer son code à 255 ordinateurs en une commande. Le vers ne se manifeste pas autrement sur le système. Cependant, ses effets sur Internet sont notables puisque tous les serveurs infectés ne

cessent de se connecter à autres machines pour envoyer le code du vers qui peut amener à un ralentissement du réseau [VIR].

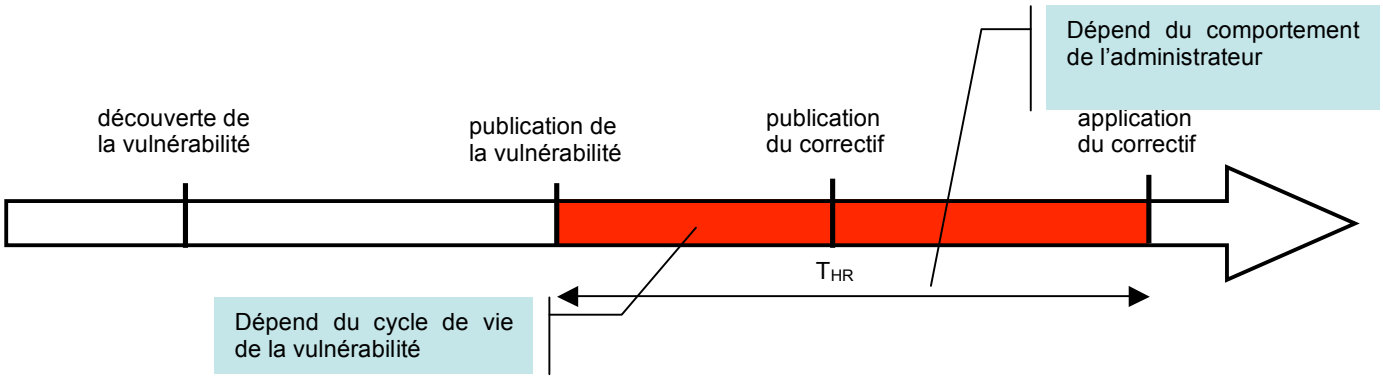


Figure 2.6 : Zone de haut risque

Avec un tel cycle de vie – c'est-à-dire avec publication de la vulnérabilité et publication du correctif simultanés – le T_{HR} a une valeur minimale de 0. Cela traduit un cycle de vie très avantageux pour l'administrateur du système dans le cadre du scénario de découverte non malveillante, ce qui est le cas de notre exemple. L'épidémie a débuté 185 jours après la publication de la vulnérabilité et du correctif. Dans ce contexte, un T_{HR} inférieur à cette valeur suffit à garantir un système sans danger vis-à-vis de cette vulnérabilité. Or, le vers Slammer a infecté un nombre très important de serveurs dès sa parution – plus de 75000 machines infectées dans les premières minutes. Cela signifie un THR supérieur à 185 jours pour de nombreux administrateurs, ce qui témoigne de l'importance de ce dernier facteur environnemental : le comportement de l'administrateur vis-à-vis des dangers liés à la sécurité.

Cette mesure est une première approche permettant de quantifier de façon très simple l'intervalle de temps durant lequel le système est potentiellement en grand danger vis-à-vis d'une vulnérabilité. Appliqué à de nombreuses vulnérabilités et sur une période de temps incluant les cycles de vie de plusieurs vulnérabilités distinctes, cette mesure peut servir de premier indicateur pour analyser la sécurité du système : l'idéal est d'avoir $T_{HR} = 0$. En effet, plus T_{HR} augmente, plus le système encourt un risque. Cette mesure prenant à la fois en considération le cycle de vie de la vulnérabilité et le comportement de l'administrateur, elle permet de mettre en évidence les manques d'un de ces deux, ou des deux, facteurs environnementaux. Prenons l'exemple suivant : une spécification de sécurité d'un système d'information est que son T_{HR} moyen ne doit pas dépasser la durée de 5 jours. Malgré un administrateur très rigoureux, le T_{HR} moyen effectif est supérieur à ce minimal de 5 jours souhaité. Cela signifie que les vulnérabilités publiées ne sont pas corrigées assez vite. Cela met en lumière que l'administrateur du système peut donc, le cas échéant, envisager d'autres outils mieux maintenus afin de maintenir son objectif de sécurité. Dans le cas extrême inverse, le laxisme de l'administrateur système peut être également mis en évidence.

Cependant, un tel indicateur ne suffit pas. D'une part, il ne prend pas en compte le comportement de la population des attaquants qui, comme nous l'avons vu, a une influence

importante. D'autre part, il est important de pouvoir considérer des mesures plus précises, que nous présentons dans le paragraphe suivant.

6 Définition des mesures envisagées

Il est nécessaire, avant de poursuivre notre étude de définir les mesures que nous comptons évaluer. Comme nous l'avons dit dans le paragraphe précédent, nous allons chercher à évaluer l'état du système vis-à-vis du phénomène d'exploitation d'une instance de vulnérabilité.

6.1 Choix de mesures déterministes ou probabilistes

La question que nous devons nous poser et le choix de mesures probabilistes ou déterministes. Pour cela, il paraît indispensable d'étudier les événements relatifs aux facteurs environnementaux afin de déterminer si leur influence peut être quantifiée de manière déterministe ou probabiliste. Pour cela, nous étudions dans un premier temps les événements caractérisant le cycle de vie de la vulnérabilité.

Nous pouvons, dans le cas d'une vulnérabilité connue, connaître de manière précise les dates du cycle de vie de la vulnérabilité ainsi que la date d'apparition du kit d'exploitation. Ainsi, comme notre approche se concentre sur les conséquence du processus d'exploitation du système, il apparaît possible de considérer ces événements comme étant des événements déterministes.

Cependant, nous considérons ces événements du point de vue du système face à l'existence d'une vulnérabilité quelconque présente sur le système. Il est donc impossible d'affirmer que le phénomène de publication de la vulnérabilité, par exemple, aura lieu à un instant précis : chaque vulnérabilité possède son cycle de vie propre. Même si les événements du cycle de vie de la vulnérabilité sont ordonnés dans le temps, leurs instants d'occurrence ne sont pas les mêmes pour deux vulnérabilités distinctes. Il en est de même pour l'instant d'apparition du kit d'exploitation.

La prise en compte de l'administrateur est elle aussi à étudier. Il est possible de connaître et de définir le comportement de l'administrateur du système que l'on considère. La fréquence à laquelle il applique des mises à jour peut être caractérisée de manière déterministe. Cependant, son comportement en cas de danger ou de compromission du système par l'exploitation de la vulnérabilité est plus incertain. De même, le temps qu'il va falloir à l'administrateur pour réparer son système après une attaque réussie est difficile à quantifier de manière déterministe. Nous choisissons donc la aussi de caractériser le comportement de l'administrateur de manière probabiliste

Nous avons vu qu'il était préférable de considérer les événements relatifs aux facteurs environnementaux de manière probabiliste. Les mesures que nous allons définir afin d'évaluer les risques encourus par le système seront donc des mesures probabilistes. Nous les présentons dans la partie suivante.

6.2 Définition des mesures

Plusieurs mesures nous apparaissent pertinentes : nous cherchons à avoir à la fois des mesures de la sécurité du système vis-à-vis de son état instantané et des mesures qui évaluent la sécurité du système considérant également les états passés du système. Ces

mesures sont des mesures de probabilité que le système ait subi ou non un événement ou une combinaison d'événements avant un instant donné. Pour les définir de manière formelle, il est tout d'abord indispensable de définir plusieurs événements, que nous définissons dans le tableau 2.2.

Compromission : le système subit une attaque couronnée de succès. Cet événement est possible s'il existe un kit d'exploitation.
Correction : l'administrateur installe le correctif de la vulnérabilité. Cet événement est possible s'il existe un correctif disponible.
Réparation : l'administrateur vérifie son système et élimine les dégâts éventuels causés par les conséquences d'une compromission.

Tableau 2.2 : Définition des événements

Nous considérons que lors d'une compromission du système, les actions de correction et de réparation citées dans le tableau 2.2 sont effectuées dans cet ordre : lorsque le système est compromis, l'administrateur va d'abord chercher à appliquer le correctif pour prévenir de nouvelles attaques avant de réparer les dégâts qui ont été causés au système. Cependant, si l'administrateur s'avère plus rapide que la population des attaquants, le phénomène de correction peut avoir lieu avant que le système n'ait été compromis.

Nous définissons les mesures qui nous intéressent dans les paragraphes suivants. Notons que ces mesures sont toutes des mesures de probabilité en fonction du temps.

6.2.1 Probabilité d'être dans un état compromis : PPC(t)

Cette mesure indique la probabilité à un instant *t* que le système ait subi une attaque couronnée de succès et soit encore dans un état où il est susceptible d'être attaqué. Elle est notée selon la définition suivante :

$$PPC(t) = P([Compromission<t] \wedge [Correction \geq t])$$

Lorsque la mesure atteint la valeur de 1, l'occurrence du phénomène de compromission du système devient une certitude.

6.2.2 Probabilité de compromission : PC(t)

Cette mesure indique la probabilité à un instant *t* que le système ait subi une attaque couronnée de succès. Cette mesure n'indique rien sur le fait que le système ait été ou non corrigé depuis. Elle est définie de la manière suivante :

$$PC(t) = P(Compromission<t)$$

6.2.3 Probabilité de compromission non réparée : PCNR(t)

Cette mesure indique la probabilité à un instant *t* que le système ait subi une attaque couronnée de succès, mais que les dégâts éventuels causés n'aient pas encore été réparés. Cette mesure est importante car elle tient compte du fait que l'attaquant a pu s'introduire sur le système et y ait gagné des privilèges qu'il puisse encore utiliser. Elle est notée de la manière suivante :

$$PCNR(t) = P([Compromission<t] \wedge [Réparation \geq t])$$

6.2.4 Probabilité de système sûr : PS(t)

Cette dernière mesure indique la probabilité à un instant *t* pour le système d'être sûr vis-à-vis de la vulnérabilité. Elle est notée PS(t) et est définie comme suit.

$$PS(t) = P([Compromission \geq t \wedge Correction<t] \vee [Compromission<t \wedge Correction<t \wedge Réparation<t])$$

Notons que cette mesure indique la probabilité pour le système d'être sûr vis-à-vis de la vulnérabilité considérée, qu'il ait, ou non, été compromis par un attaquant ayant exploité la vulnérabilité.

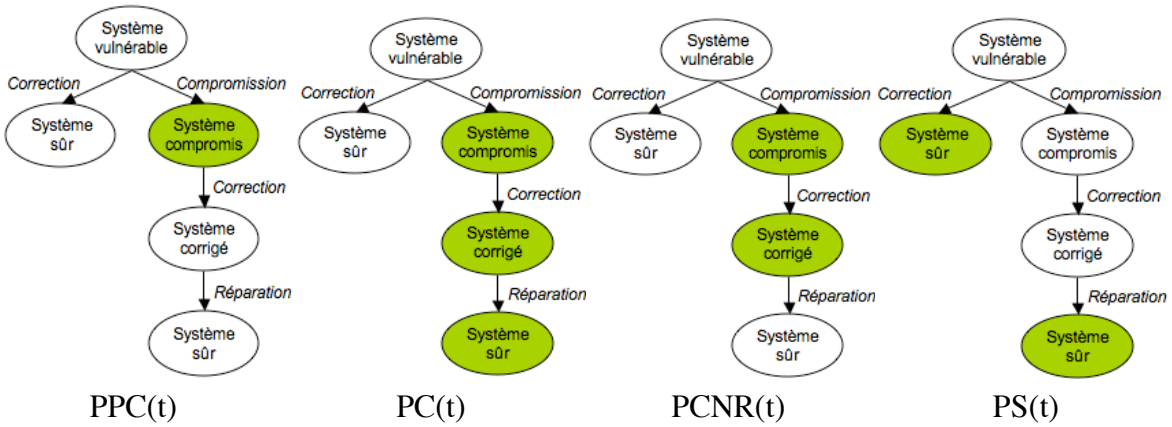


Figure 2.7 : Lien entre mesures et états du système

Les quatre mesures définies sont illustrées dans la figure 2.7. A partir de l'état vulnérable du système, deux scénarios sont possibles : soit l'administrateur corrige le système avant une intrusion, soit un attaquant exploite la vulnérabilité avant qu'elle ne soit corrigée. Dans le second cas, le système compromis est corrigé puis réparé. Pour chaque mesure, les états considérés sont colorisés dans le schéma. Ces mesures sont des mesures basées sur une échelle absolue. Afin de parvenir à leur évaluation, il est indispensable maintenant de passer au stade de la modélisation. Une fois cette étape de modélisation effectuée, nous détaillerons comment nous associons les mesures avec des contrôles sur les places du modèle.

7 Conclusion

Dans ce chapitre, nous avons fait une mise au point importante sur les vulnérabilités que nous pouvions inclure dans une approche comme la nôtre. Nous avons fait le choix de considérer les vulnérabilités issues de la phase de développement du système et de laisser de côté celles qui sont liées à la phase d'utilisation du système. A partir de ce choix, nous avons considéré l'environnement du système qui pouvait avoir une influence significative sur le danger lié à la présence de la vulnérabilité sur le système. Nous avons identifié trois paramètres – le cycle de vie de la vulnérabilité, la mise au point du kit d'exploitation par la population d'attaquants et le comportement de l'administrateur vis-à-vis de la sécurité de son système et de l'application des correctifs. Nous avons également étudié et mis en évidence les différentes interactions entre ces facteurs environnementaux. En les étudiant, nous avons fait le choix d'un paramètre d'étude qui est le comportement de l'administrateur. Ce choix a pour but de pouvoir mettre au point une mesure de rapport de coût entre coût de maintenance du niveau de sécurité et niveau de sécurité exige ainsi que de pouvoir étendre notre approche à une plage de vulnérabilités plus étendue par la suite. Enfin, nous avons vu dans notre chapitre une première mesure simple quantifiant facilement le temps durant lequel le système court un danger significatif compte tenu de deux des trois facteurs environnementaux que nous avons identifiés. Cette mesure peut être utilisée comme mesure de base pour quantifier une zone de risque moyenne dans laquelle se trouve le système pour l'ensemble des vulnérabilités auxquels celui-ci est confronté. Elle met de plus en évidence les besoins d'une évaluation plus poussée. Nous avons donc défini les mesures précises que nous envisageons avant de passer à l'étape de modélisation du processus d'exploitation d'un vulnérabilité.

Le chapitre suivant décrit l'étape d'élaboration du modèle permettant l'évaluation des mesures présentées. Nous y présentons également les résultats des expériences menées pour la validation de notre approche.

Chapitre 3 : Modélisation et principe de mesure

Nous avons identifié trois facteurs environnementaux qui ont une influence significative sur le processus de compromission d'un système d'information par l'exploitation de vulnérabilités issues de la phase de conception et de développement. Notre objectif étant la mise au point de mesures quantitatives de la sécurité en considérant l'environnement du système et son évolution, nous abordons la prise en compte de ces facteurs dans notre approche.

Les différentes approches existantes que nous avons rencontrées se basent sur des positionnements très différents. Il nous faut dans un premier temps identifier le point de vue que nous allons adopter afin d'incorporer les facteurs environnementaux dans une approche capable de produire des mesures quantitatives et répondant aux deux objectifs de mesures que nous nous sommes fixés : 1) la mesure de la distance à un objectif et 2) la mesure de risque en considérant une situation établie. Il nous faut ensuite identifier la formalisation à même de satisfaire les spécifications de notre approche. Une fois ces décisions prises, nous pourrons nous pencher sur la phase de modélisation en elle-même. Cette modélisation devra tenir compte des facteurs environnementaux que nous avons identifiés dans le chapitre précédent y compris de leur évolution dans le temps et de leurs dépendances.

1 Choix de l'approche : une vulnérabilité par modèle

Comme nous l'avons vu dans le chapitre précédent, nous nous concentrons sur une approche de l'évaluation de la sécurité considérant l'environnement du système et les vulnérabilités de la phase de développement. Reste à déterminer le point de vue que nous allons adopter. En effet, les facteurs de l'environnement que nous avons identifiés sont complexes et nous devons nous assurer de ne pas produire une modélisation qui pourrait s'avérer inutilisable et/ou susceptible de produire des résultats biaisés. Nous rappelons le système complet que nous comptons considérer dans la figure 3.1 : le système d'information avec l'ensemble de ses vulnérabilités issues de la phase de conception et implémentation, l'ensemble de la population d'attaquants ainsi que l'administrateur du système. Tous ces éléments réunis forment un système complexe et ce paragraphe étudie le point de vue qu'il nous sera possible d'adopter.

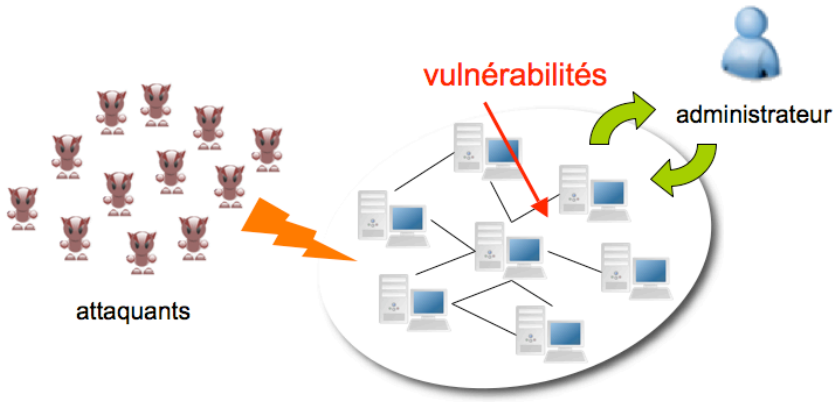


Figure 3.1 : Le système et son environnement

Le système informatique que nous voulons évaluer possède a priori, comme nous l'avons évoqué, plusieurs vulnérabilités issues de la phase de conception et d'implémentation. Chaque vulnérabilité possède un cycle de vie qui lui est propre. Il apparaît donc assez difficile de considérer globalement l'ensemble des vulnérabilités si nous tenons à les caractériser avec ce point de vue. Une idée possible serait donc de débuter notre approche en considérant une vulnérabilité, puis d'étudier par la suite une façon de rendre notre approche plus globale.

Nous avons également identifié la population des attaquants comme un facteur environnemental qui évolue dans le temps. Nous considérons donc les attaquants de manière globale sans distinguer les différents niveaux de compétence que nous pouvons rencontrer. A l'inverse, nous avons rencontré plusieurs approches qui prennent en considération un seul attaquant face au système. Ce type d'approche permet de caractériser l'attaquant de façon précise et ainsi de tenir compte d'un ensemble de caractéristiques importantes comme le mobile de l'attaquant ou encore son degré de compétence. Cependant, ayant réussi à caractériser la population d'attaquants par l'événement important qu'est l'apparition du kit d'exploitation, il nous semble possible de prendre en considération toute la population des attaquants de façon globale. Ceci est également rendu possible par le fait que nous nous intéressons à une seule vulnérabilité à la fois.

Enfin, le comportement de l'administrateur reste le facteur local important que nous allons prendre en considération. Comme nous avons fait l'hypothèse d'un seul administrateur – ou d'une seule équipe d'administrateurs travaillant de concert, son action ne pose pas de question particulière de modélisation.

Nous ne considérons dans cette approche qu'une instance de vulnérabilité à la fois. Cela permet de manipuler un modèle peu complexe tout en prenant toujours en considération l'ensemble des facteurs environnementaux. Il nous faut maintenant trouver le formalisme de modélisation qui conviendra à notre point de vue. C'est l'étude que nous présentons dans le paragraphe suivant.

2 Choix de représentation

Dans cette partie, nous présentons et justifions le choix du formalisme de modélisation que nous utiliserons dans notre approche. Dans une première partie, nous analysons nos besoins et déterminons quel est le formalisme de modélisation le plus adapté. Dans une seconde partie, nous détaillons le fonctionnement du formalisme choisi afin de ne pas perdre le lecteur lorsque nous présenterons la modélisation en détail.

2.1 Description des besoins et choix du formalisme

Pour notre modélisation, nous avons plusieurs besoins. Ceux-ci vont déterminer quel sera le formalisme de modélisation que nous utiliserons. Nous avons, dans un premier temps, procédé à l'identification des formalismes les plus utilisés dans le domaine de la sûreté de fonctionnement. Nous avons identifié trois formalismes importants :

- Les chaînes de Markov : elles permettent la modélisation d'un processus stochastique tel que l'état actuel dépend uniquement de l'état précédent ;
- Les réseaux de Pétri stochastiques généralisés [MAR 95] : ils sont une extension des réseaux de Pétri stochastiques. Ils permettent de prendre en considération des événements déterministes immédiats et des événements probabilistes modélisés par des distributions exponentielles. Ils sont facilement transformables en chaînes de Markov ;
- Les SAN (*Stochastic Activity Networks* en anglais) [SAN 02] : ils sont une extension des réseaux de Pétri stochastiques généralisés mais peuvent modéliser des événements caractérisés par des distributions de probabilité quelconques.

Un de nos besoins est la nécessité de posséder un outil disponible pour la résolution quantitative de notre modèle. Dans le cadre de la sûreté de fonctionnement, deux outils sont utilisés : SURF-2 et Möbius.

- Le logiciel SURF-2, développé au LAAS-CNRS, a été mis au point pour la résolution de modèles markoviens [BEO 93]. Cet outil présente l'avantage de fournir les mesures de la sûreté de fonctionnement décrites dans [LAP 95]. Il prend en entrée soit une chaîne de Markov, soit un réseau de Pétri stochastique généralisé.
- L'outil Möbius a été quant à lui développé pour la résolution des modèles SAN [CLA 01]. Il a été au départ développé pour le calcul de propriétés liées à la sûreté de fonctionnement comme la fiabilité ou la disponibilité ou encore le calcul de performances. Mais le large spectre d'utilisation des SAN a conduit l'outil Möbius a être utilisé dans des domaines divers comme l'aéronautique ou la biologie.

Le choix de l'outil dépendra donc du choix du formalisme de modélisation associé.

Notre second besoin est de pouvoir produire une modélisation dont les événements peuvent être modélisés suivant des distributions de probabilité aussi diverses que possibles. En effet, à ce stade de notre étude, nous ne sommes pas capables de dire quelles seront les distributions de probabilité susceptibles de modéliser les événements de notre modèle. Surtout, il nous est difficile de pouvoir garantir que ces événements pourront être modélisés par des distributions exponentielles. Nous faisons donc le choix de modéliser notre approche par des SAN. Selon la modélisation à laquelle nous aboutirons, il sera possible de transformer notre modèle SAN en réseaux de Pétri stochastiques généralisés, plus contraignant du fait des distributions de probabilité utilisables et des conditions de sensibilisation des activités.

Dans le paragraphe suivant, nous rappelons les divers éléments qui peuvent composer un SAN. Nous détaillons leur fonctionnement et leur rôle dans la modélisation afin de faciliter la compréhension des futurs modèles au lecteur.

2.2 Rappel et description du formalisme des SAN

Un SAN est composé de quatre types d'éléments différents :

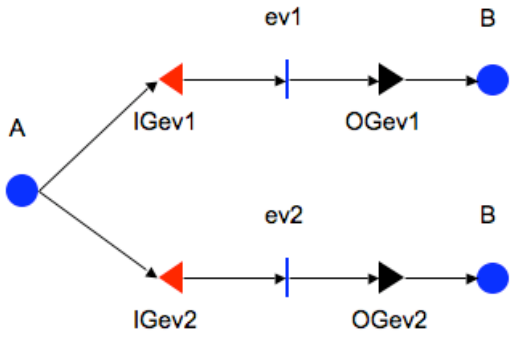
- Les **places** : elles traduisent l'état partiel du système modélisé et sont représentées par des cercles. Elles peuvent contenir un ou plusieurs jetons.
- Les **activités** : elles modélisent les événements qui changent l'état du système modélisé. Elles peuvent être instantanées ou temporisées. Les activités instantanées sont représentées par une barre. Les activités temporisées sont représentées par un rectangle. Les activités temporisées peuvent être déterministes ou probabilistes. Les événements modélisés par activités déterministes sont déclenchés à un instant précis après la sensibilisation de l'activité. Les activités probabilistes sont régies par une distribution de probabilité paramétrée.
- Les **portes d'entrée** : elles contiennent les pré-conditions nécessaires à la sensibilisation de l'activité. Ces pré-conditions sont représentées sous la forme d'une expression logique. Si les conditions sont vérifiées, l'activité est sensibilisée. Les portes d'entrée peuvent aussi avoir des actions sur la transition ou la production des jetons dans les places. Elles sont représentées par des triangles. Dans ce mémoire, elles seront associées à la couleur rouge.
- Les **portes de sortie** : elles contiennent les conséquences suite à la transition de l'activité. Ces conséquences peuvent déplacer ou produire un ou plusieurs jetons dans les places du SAN. Elles sont représentées par des triangles. Dans ce mémoire, elles seront associées à la couleur noire pour faciliter la distinction entre portes d'entrée et portes de sortie.

Dans un SAN, comme dans les réseaux de Pétri, plusieurs activités peuvent être sensibilisées au même moment : deux activités instantanées sensibilisées au même moment sont en conflit. Cela signifie qu'il est impossible de savoir quel événement se produira avant l'autre. Cependant, une activité instantanée sensibilisée sera prioritaire devant les activités temporisées sensibilisées. La figure 3.2 représente trois exemples de SAN.

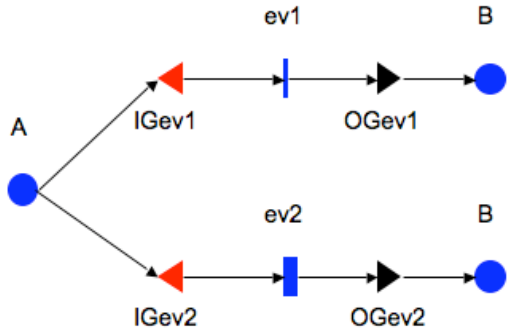
Il est à noter que les places seront nommées avec la notation suivante : *NomPlace* ; les activités seront notées comme suit : *nomActivite* ; enfin les portes d'entrées porteront un nom caractérisé par le nom de l'activité associée et par le préfixe *IG* (pour "input gate" en anglais). Il en sera de même pour les portes de sortie qui porteront un nom caractérisé par le nom de l'activité associée et par le préfixe *OG* (pour "output gate" en anglais).

Dans le premier SAN (a), les activités *ev1* et *ev2* sont deux activités instantanées. Elles sont en conflit si elles sont toutes deux sensibilisées. Dans le second SAN (b), l'activité *ev1* est instantanée et l'activité *ev2* est temporisée. Si on considère que les deux activités sont sensibilisées, l'activité *ev1* est prioritaire sur l'activité *ev2*. Enfin, dans le troisième SAN (c), les deux activités *ev1* et *ev2* sont temporisées. Elles sont concurrentes mais ne sont pas en conflit : les distributions de probabilité associées aux activités décident du comportement du système et de la probabilité d'occurrence de l'événement *ev1* avant ou après l'événement *ev2*.

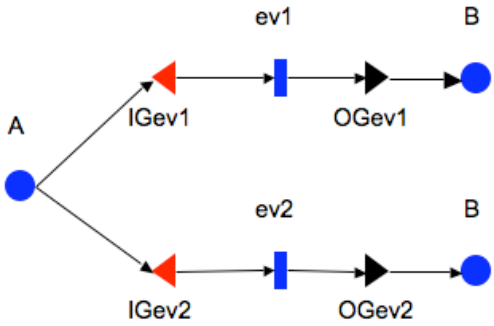
Il est également à préciser que la pré-condition de sensibilisation d'un événement peut tenir compte du marquage d'une place sans pour autant le modifier. Cette propriété peut être très utile pour tester l'état partiel du système modélisé.



(a)



(b)



(c)

Figure 3.2 : Exemples de SAN

Les modèles développés en SAN peuvent également être composés : des places peuvent ainsi être partagées entre plusieurs modèles SAN, qualifiés d'atomiques. Cela se fait par l'utilisation d'un opérateur JOIN qui permet de synchroniser des modèles indépendants et de gérer les places partagées par les modèles atomiques. Mais n'utilisant pas ces propriétés dans l'immédiat, nous reviendrons sur leur utilisation dans le chapitre 5.

Nous allons donc utiliser les SAN pour produire la modélisation du processus d'exploitation d'une vulnérabilité par une population d'attaquants. Nous expliquons cette modélisation dans la partie suivante.

3 Modélisation du processus d'exploitation d'une vulnérabilité

La première étape de notre modélisation est donc de mettre au point un modèle représentant le processus de compromission du système par l'exploitation d'une vulnérabilité. Nous devons intégrer dans notre modèle les facteurs environnementaux que nous avons identifiés dans le chapitre précédent, leur évolution dans le temps mais également les conséquences de leurs interdépendances. Cette partie détaille donc la composition de ce modèle.

3.1 Le cycle de vie de la vulnérabilité

Nous nous intéressons dans un premier temps à la modélisation du cycle de vie de la vulnérabilité. Le fait que nous ayons défini et étudié le cycle de vie comme un ensemble d'événements ordonnés dans le temps va permettre de faciliter sa modélisation. Les événements du cycle de vie, pris indépendamment des autres facteurs environnementaux, se modélisent donc facilement par une suite de places et d'activités, accompagnées des portes d'entrée et de sortie associées à chaque activité, tel que cela est présenté dans la figure 3.3.

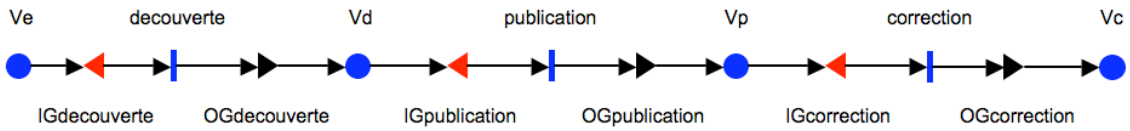


Figure 3.3 : Modélisation du cycle de vie de la vulnérabilité

Les quatre phases du cycle de vie de la vulnérabilité sont modélisées par quatre places : *Ve*, *Vd*, *Vp*, et *Vc*. Les activités *decouverte*, *publication*, et *correction* traduisent les événements du cycle de vie.

La place *Ve* modélise le fait que la vulnérabilité existe. C'est l'état qui précède la découverte. Cela veut dire que la vulnérabilité existe et que le composant vulnérable est diffusé. Cependant, à ce stade, personne n'a connaissance de l'existence de la vulnérabilité jusqu'à l'instant de la découverte. Ce phénomène de découverte est modélisé par l'activité *decouverte*. Cette activité est entourée, comme chaque activité, d'une porte d'entrée et d'une porte de sortie. La porte d'entrée *IGdecouverte* ne vérifie que la présence d'un jeton dans la place *Ve*. Une fois l'activité *decouverte* franchie, la porte de sortie *OGdecouverte* retire un jeton à la place *Ve* et ajoute un jeton à la place *Vd*.

La place *Vd* représente l'état où la vulnérabilité est découverte. Depuis cet état, la vulnérabilité peut être publiée. Cette action est traduite par l'activité *publication*. La porte d'entrée *IGpublication* vérifie la présence d'un jeton dans la place *Vd*. La porte de sortie *OGpublication* soustrait un jeton de la place *Vd* et ajoute un jeton à la place *Vp*.

La place *Vp* traduit le fait que la vulnérabilité est publiée. Cet état va conduire à l'action de publication d'un correctif de la vulnérabilité, modélisé par l'activité *correction*. La porte d'entrée *IGcorrection* vérifie la présence d'un jeton dans la place *Vp*. La porte de sortie *OGcorrection* applique les conséquences de l'activité *correction* : un jeton est soustrait de la place *Vp* et en ajoute un dans la place *Vc*, qui modélise ainsi l'état où la vulnérabilité est découverte, publiée et où il existe un correctif disponible.

Il est à noter que cette modélisation ne tient pas encore compte des dépendances que nous avons identifiées dans le chapitre précédent. Notre première démarche est de modéliser chaque facteur environnemental aussi indépendamment que possible des autres facteurs. Nous verrons par la suite comment la modélisation incorporera les dépendances que nous avons identifiées.

3.2 La modélisation du comportement de l'attaquant

Le second facteur environnemental que nous avons à modéliser est le comportement de l'attaquant. Nous avons caractérisé celui-ci par un événement déterminant dans le processus d'exploitation de la vulnérabilité qui est la mise au point du kit d'exploitation. Nous modélisons donc facilement ce facteur environnemental comme cela est illustré sur la figure 3.4.

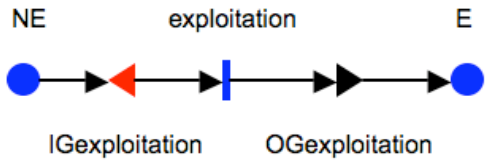


Figure 3.4 : Modélisation du comportement des attaquants

Deux places et une activité suffisent à modéliser la mise au point du kit d'exploitation. La place *NE* traduit le fait que la population d'attaquants n'a pas encore mis au point de kit d'exploitation. La place *E* indique, en revanche, l'existence d'un kit d'exploitation disponible à la population des attaquants. Le passage de l'état traduit par la place *NE* à celui traduit par la place *E* est faite par l'activité *exploitation*. La porte d'entrée de cette activité, nommée *IGexploitation*, contrôle la présence d'un jeton dans la place *NE*. La porte de sortie, nommée *OGexploitation*, soustrait un jeton de la place *NE* et en ajoute un dans la place *E*.

Il est à noter que l'action directe de l'attaquant sur le système, c'est-à-dire le processus d'attaque en lui-même, est considérée du point de vue du système et non de l'attaquant. Nous le modéliserons donc lorsque nous considérerons l'état du système (cf. paragraphe 3.4).

3.3 Modélisation des interdépendances entre cycle de vie et comportement de l'attaquant

Dans le chapitre précédent, nous avons étudié chaque facteur environnemental et mis en évidence des dépendances très fortes entre le cycle de vie de la vulnérabilité et le comportement de la population d'attaquants. Nous avons isolé deux scénarios dépendants de l'origine de la découverte de la vulnérabilité. Nous choisissons donc de séparer les deux scénarios pour créer deux modèles correspondants aux deux scénarios envisagés.

3.3.1 Scénario de découverte non malveillante

Considérons en premier lieu le scénario de découverte non malveillante. Cette hypothèse implique que la population d'attaquants n'a pas connaissance de l'existence de la vulnérabilité avant l'événement de publication. Nous modélisons ces interdépendances dans le modèle illustré par la figure 3.5.

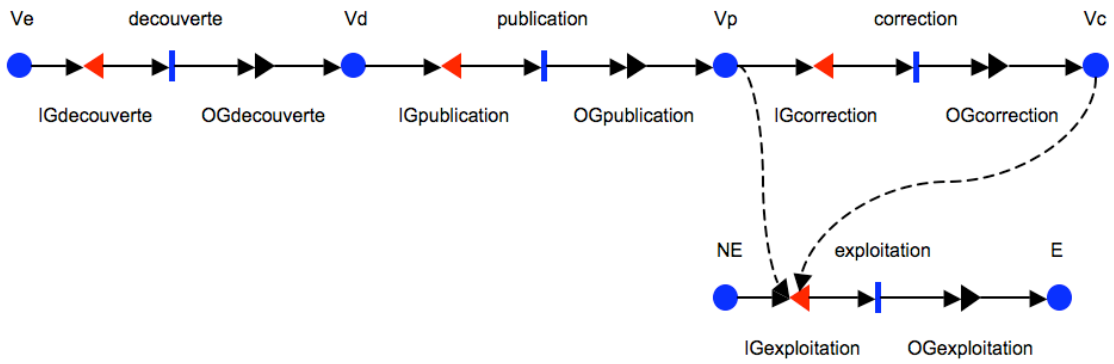


Figure 3.5 : Modélisation des interdépendances entre cycle de vie et comportement de l'attaquant avec l'hypothèse d'une découverte d'origine non malveillante

Le cycle de vie de la vulnérabilité reste le même. Cependant, il y a l'ajout d'une condition de passage de l'activité *exploitation*. Cette condition est la vérification de la présence d'un jeton dans les places *Vp* ou *Vc*. Elle permet de valider que la vulnérabilité a été publiée : c'est-à-dire qu'elle est, soit publiée, soit publiée et possédant un correctif disponible. Les arcs ajoutés au modèle sont dessinés en pointillés sur la figure 3.5. Ils indiquent que la pré-condition de la porte d'entrée *IGexploitation* nécessite de connaître le marquage des places *Vp* et *Vc*. Il est à noter que la présence de ces arcs n'est pas obligatoire. En effet, comme nous l'avons vu précédemment, la porte d'entrée *IGexploitation* permet de gérer la vérification de la présence de jetons dans une ou plusieurs places par une pré-condition de sensibilisation de l'activité *exploitation*. Elle contient donc la pré-condition logique suivante.

[marquage(NE) == 1] ET [[marquage(Vp) == 1] OU [marquage(Vc) == 1]]

Il est important de préciser que la fonction *marquage* utilisée dans la formule retourne le nombre de jetons de la place indiquée en paramètre.

3.3.2 Scénario de découverte malveillante

Considérons maintenant le scénario de découverte malveillante. Avec cette hypothèse, nous devons prendre en considération que la population d'attaquants connaît l'existence de la vulnérabilité dès la découverte. La mise au point du kit d'exploitation peut donc se faire dès cet instant là. Le modèle ainsi obtenu est illustré sur la figure 3.6.

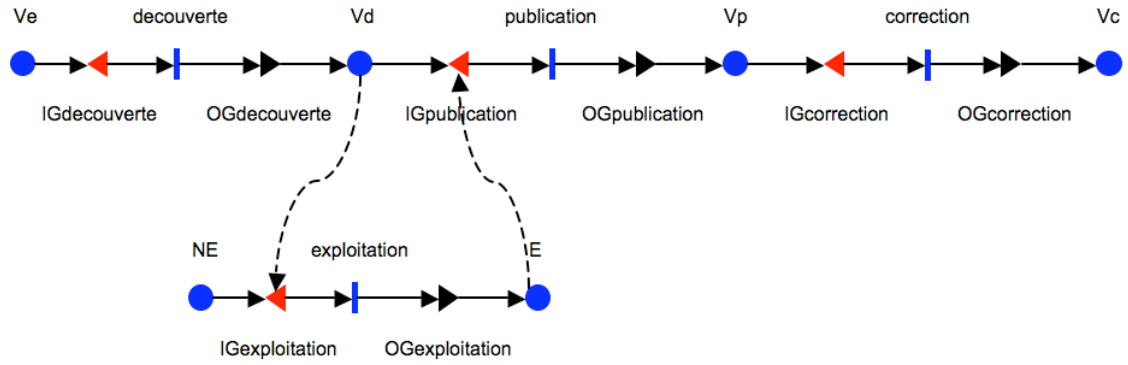


Figure 3.6 : Modélisation des interdépendances entre cycle de vie et comportement de l'attaquant avec l'hypothèse d'une découverte d'origine malveillante

Ici encore, les arcs modélisant les dépendances ont été ajoutés en pointillés car il n'est pas nécessaire de les faire apparaître explicitement dans le modèle. La première dépendance concerne l'événement de mise au point du kit d'exploitation : la vulnérabilité doit avoir été découverte pour qu'un kit d'exploitation puisse être mis au point. Cette dépendance est modélisée par l'arc allant de la place *Vd* à la porte d'entrée *IGexploitation*. La pré-condition de la porte d'entrée *IGexploitation* devient donc :

[marquage(NE) == 1] ET [marquage(Vd) == 1]

La seconde dépendance concerne l'événement de publication de la vulnérabilité. C'est la mise au point du kit d'exploitation, et surtout son utilisation par la population des attaquants qui entraînera la publication de la vulnérabilité. Cette dépendance est modélisée par l'arc allant de la place *E* à la porte d'entrée *IGpublication*. Ainsi, la pré-condition de la porte d'entrée *IGpublication* est donc :

[marquage(Vd) == 1] ET [marquage(E) == 1]

Grâce à l'enrichissement des pré-conditions des événements du cycle de vie et de la mise au point du kit d'exploitation, nous avons pu modéliser les interdépendances fortes

entre ces deux facteurs environnementaux. Il nous reste maintenant à modéliser le troisième facteur de l'environnement ainsi que l'état du système d'information lui-même.

3.4 Modélisation de l'état du système et de l'action de l'administrateur

Le troisième facteur environnemental que nous avons identifié est l'action de l'administrateur sur le système. Contrairement aux deux autres facteurs environnementaux que nous venons de modéliser, le comportement de l'administrateur est un facteur environnemental local qui est donc propre au système. Ce dernier va donc agir en fonction de l'état du système. Il est donc difficile de séparer la modélisation du comportement de l'administrateur de la modélisation de l'état du système lui-même.

Certains états du système sont très importants pour évaluer les mesures que nous avons énoncées dans le chapitre précédent. Dans cette partie, nous présentons d'abord ces états du système avant de détailler la modélisation.

3.4.1 Description des états fondamentaux

Le système s'avère en danger quand la vulnérabilité est présente dans le système et qu'il existe un kit d'exploitation permettant un attaque. Cet état est l'état que nous appelons **exposé**. Lorsque le système est dans cet état, deux actions sont possibles (figure 3.7) :

- soit l'administrateur peut appliquer le correctif de la vulnérabilité dans l'hypothèse où celui-ci est disponible : le système devient **sûr** vis-à-vis de la vulnérabilité.
- soit un attaquant exploite la vulnérabilité avec succès et réussit ainsi à compromettre le système, qui passe dans l'état **compromis**.

Lorsqu'il se trouve dans l'état compromis, la seule action possible est l'application du correctif par l'administrateur, ce qui amène le système dans l'état **corrigé**. Cet état signifie que le système est protégé vis-à-vis de la vulnérabilité mais les dégâts éventuels causés par une intrusion n'ont pas été réparés. Il reste donc une étape de réparation à effectuer pour rendre le système sûr vis-à-vis de la vulnérabilité.

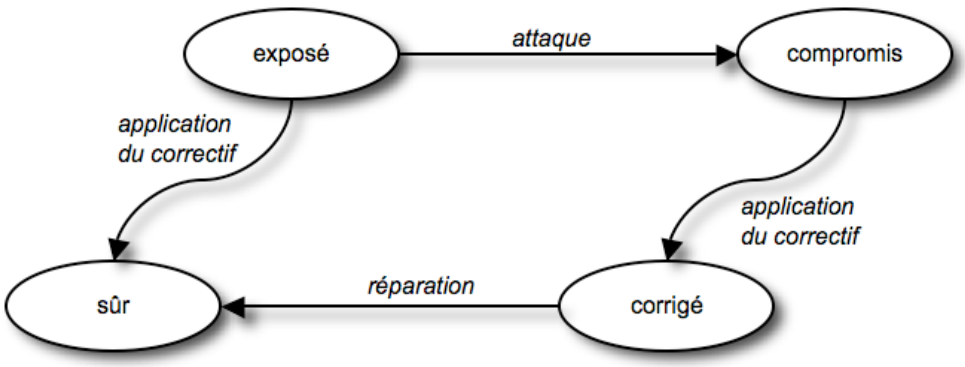


Figure 3.7 : Etats du système en présence d'une vulnérabilité et de l'existence d'un kit d'exploitation

3.4.2 Détail de la modélisation

Pour détailler les états du système, nous suivons un scénario classique de compromission du système par exploitation de la vulnérabilité.

La première étape est l'installation du composant vulnérable sur le système. Cet événement est important car nous pouvons être confrontés à plusieurs situations qu'il nous faut considérer : la vulnérabilité peut être dans un composant installé depuis longtemps – par exemple un logiciel installé par défaut sur le système. Dans ce contexte, l'événement d'installation du composant vulnérable peut être très antérieur au phénomène de découverte de la vulnérabilité.

A l'inverse, la vulnérabilité peut être celle d'un composant tel qu'une mise à jour du système ou d'un de ses logiciels. Ainsi, le cycle de vie est susceptible d'avoir évolué bien avant l'installation du composant vulnérable. Cet événement est modélisé par l'activité *install*. L'état précédant cette activité est l'état où le système n'est pas en danger. Il est modélisé par la place *ok*. L'état résultant de cette activité est l'état vulnérable, modélisé par la place *Vulnerable*. Cet état traduit le fait que le système possède la vulnérabilité. Il passe dans l'état exposé, modélisé par la place *Expose*, si un kit d'exploitation existe pour cette vulnérabilité : le système est alors réellement en danger car il est susceptible de subir une attaque. Ce changement d'état est traduit par l'activité instantanée *test*. La pré-condition contenue dans la porte d'entrée *IGtest* vérifie la présence d'un jeton dans les places *E* et *Vulnerable*. La porte de sortie *OGtest* soustrait un jeton de la place *Vulnerable* et en ajoute un à la place *Expose*. Notons que cette activité n'est pas un événement mais un test effectué pour les besoins de la modélisation.

Dans l'état exposé, le système est susceptible de subir une attaque. Si celle-ci a lieu, le système est compromis. L'occurrence du phénomène d'attaque, rendue possible grâce à l'existence du kit d'exploitation, peut être plus ou moins rapide selon la phase du cycle de vie de la vulnérabilité. En effet, la probabilité d'attaque ne va pas être la même selon que la vulnérabilité soit nouvelle ou ancienne, juste publiée ou déjà corrigée depuis longtemps. Les événements du cycle de vie de la vulnérabilité et le temps écoulé depuis ces événements sont deux éléments importants qu'il faut considérer dans la modélisation du phénomène d'attaque. Séparer ce phénomène en trois activités permet de prendre en considération les événements du cycle de vie. Le temps écoulé entre ces événements sera pris en compte lors de la modélisation de ces événements distincts. Ainsi, nous modélisons le phénomène d'attaque par plusieurs activités parallèles nommées *attaqueVd*, *attaqueVp*, et *attaqueVc*. Elles correspondent aux phénomènes d'attaque durant la phase où la vulnérabilité est découverte, publiée ou avec un correctif disponible (cf. figure 3.8). Pour chacune de ces activités, une porte d'entrée vérifie le marquage des places *Expose* et *Vd*, *Vp* ou *Vc* selon que la vulnérabilité soit respectivement découverte, publiée, ou ayant un correctif disponible. Les trois prédicats des portes d'entrée *IGattqueVd*, *IGattaqueVp*, et *IGattaqueVc* sont détaillés dans le tableau 3.1.

Les portes de sortie *OGattaqueVd*, *OGattaqueVp* et *OGattaqueVc* gèrent la transition du jeton de la place *Expose* à la place *Compromis*. Cette place modélise l'état compromis résultant d'une attaque réalisée avec succès : c'est l'état que nous voulons épargner au système.

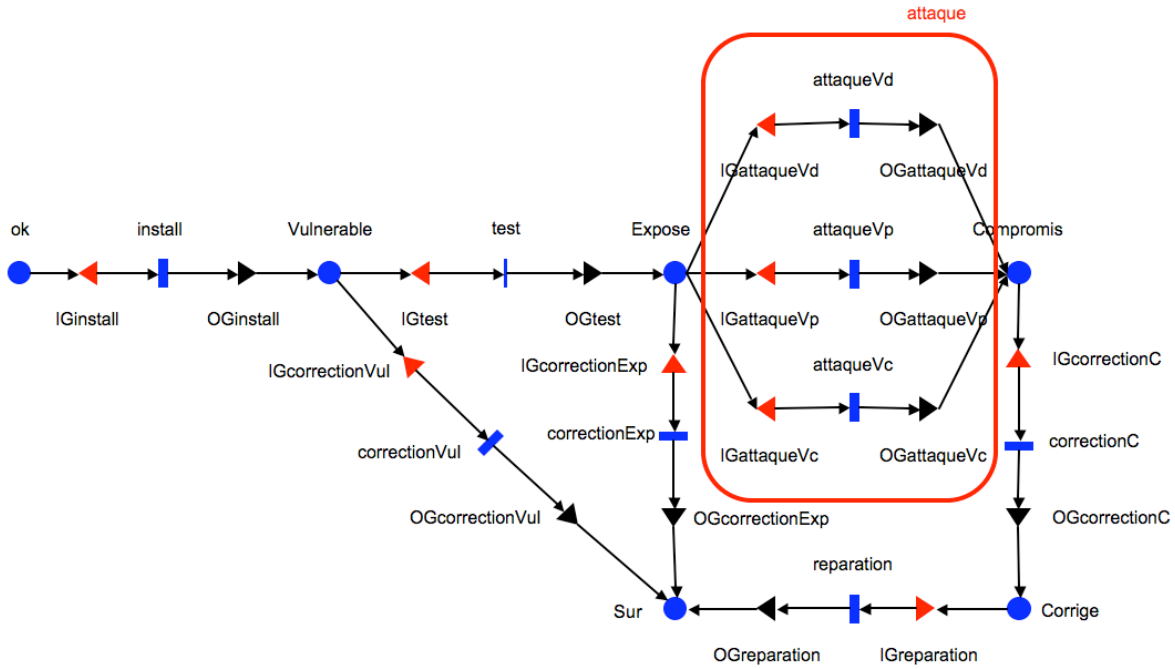


Figure 3.8 : Modélisation de l'état du système et du comportement de l'attaquant

Porte d'entrée	Prédicat
IGattaqueVd	(marquage(E)==1) ET (marquage(Vd) == 1) ET (marquage(Expose) == 1)
IGattaqueVp	(marquage(E)==1) ET (marquage(Vp) == 1) ET (marquage(Expose) == 1)
IGattaqueVc	(marquage(E)==1) ET (marquage(Vc) == 1) ET (marquage(Expose) == 1)

Tableau 3.1 : Prédicats des portes d'entrée des activités modélisant le phénomène d'attaque

Nous avons fait l'hypothèse que seule l'application du correctif peut permettre de ramener le système dans un état corrigé. Une fois le système dans un état compromis, l'application du correctif devient donc la seule action possible. L'état du système ainsi obtenu signifie que la vulnérabilité a été masquée ou corrigée. Le système n'est donc plus vulnérable. L'événement d'application du correctif est modélisé par l'activité *correctionC*. Cette activité possède une porte d'entrée *IGcorrectionC* qui vérifie que la vulnérabilité possède un correctif disponible par la présence d'un jeton dans la place *Vc*. La porte de sortie *OGcorrectionC* enlève un jeton à la place *Compromis* et en ajoute un à la place *Corrige*. Cependant, les dégâts dus à l'attaque et à l'intrusion de l'attaquant ne sont pas encore réparés. Pour atteindre l'état sûr, modélisé par la place *Sur*, il faut encore réparer les dégâts causés par l'attaque. Cette tache est modélisée par l'activité *reparation*. La porte d'entrée *IGreparation* n'a pas de pré-condition particulière, mis à part la présence d'un jeton dans la place *Corrige*.

L'administrateur ne réagit parfois pas assez vite pour appliquer le correctif avant que le système ait été compromis. Mais dans le cas d'un administrateur conscient des

enjeux de la sécurité du système, celui-ci peut, s'il a connaissance de la vulnérabilité, appliquer le correctif durant les phases où le système est 1) dans l'état vulnérable ou bien 2) dans l'état exposé. Cette action nécessite, bien sûr, que le correctif soit disponible. Dans la modélisation, une première activité, nommée *correctionVul*, modélise l'événement d'application du correctif lorsque le système est vulnérable. La porte d'entrée *IGcorrectionVul* vérifie la présence d'un jeton dans les places *Vulnerable* et *Vc*. La porte de sortie *OGcorrectionVul* gère la transition du jeton de la place *Vulnerable* à la place *Sur*. L'activité *correctionExp* modélise l'événement d'application du correctif de la vulnérabilité lorsque le système est dans un état exposé. Une porte d'entrée *IGcorrectionExp* vérifie la présence de jetons dans les places *Expose* et *Vc*. La porte de sortie *OGcorrectionExp* gère la transition du jeton de la place *Expose* à la place *Sur*. Les prédicats des trois portes d'entrées des activités modélisant l'application du correctif sont précisés dans le tableau 3.2.

Porte d'entrée	Prédicat
IGcorrectionVul	(marquage(Vc)==1) ET (marquage(Vulnerable) == 1)
IGcorrectionExp	(marquage(Vc)==1) ET (marquage(Expose) == 1)
IGcorrectionC	(marquage(Vc)==1) ET (marquage(Compromis) == 1)

Tableau 3.2 : Prédicats des portes d'entrée des activités modélisant l'application du correctif

Le comportement de l'administrateur se traduit donc par les trois actions d'application du correctif selon que le système est vulnérable, exposé ou compromis, mais également par le nettoyage et la réparation du système lorsque celui-ci a été compromis et a subi une intrusion. Cette dernière tâche a cependant une importance relative par rapport à l'action d'application du correctif car celle-ci peut être une action préventive si le correctif est appliqué avant l'attaque.

3.5 Modélisation des deux scénarios

Dans les paragraphes précédents, nous avons montré comment modéliser l'état du système en présence d'une vulnérabilité et en tenant compte des trois facteurs environnementaux. Nous avons découpé notre démarche et aboutis à trois modélisations SAN interdépendantes. Les figures 3.9 et 3.10 présentées dans cette partie montre les deux modélisations correspondant aux deux scénarios d'origine de découverte de la vulnérabilité.

La figure 3.9 représente le modèle correspondant au scénario de découverte d'origine non malveillante. Le cycle de vie de la vulnérabilité et la mise au point du kit d'exploitation sont tels qu'ils ont été décrits dans le paragraphe 3.3.1. Les flèches ajoutées en pointillés indiquent la présence de la vérification de la présence d'un jeton dans le prédicat de la porte d'entrée désignée. Il est à noter que le phénomène d'attaque n'est modélisé que par deux activités : selon ce scénario, il est impossible que la population des attaquants ait mis au point un kit d'exploitation – et donc mené une attaque contre la système – tant que la vulnérabilité n'a pas été publiée.

Dans la figure 3.10, les places vérifiées dans les prédicats des portes d'entrée sont également reliées à ces dernières par une flèche pointillée. Il est important de remarquer la

présence de l'activité d'attaque *attaqueVd* qui n'apparaît pas dans le modèle précédent. Cela est dû au scénario de découverte malveillante qui permet à la population d'attaquant d'attaquer le système en phase de découverte dès lors que le kit d'exploitation a été mis au point.

Nous avons mis au point deux modèles correspondant aux deux scénarios basés sur l'origine de la découverte de la vulnérabilité. Il paraît important, à ce stade, de vérifier la validité de notre démarche. Le paragraphe suivant montre comment nous nous sommes basés sur des données réelles pour paramétrer notre modèle.

4 Définition des mesures dans le cadre du modèle

Dans le chapitre précédent, nous avons défini les mesures qui nous semblaient pertinentes. Nous allons voir dans ce paragraphe comment nous les exprimons dans le cadre du modèle.

PPC(t) mesure la probabilité à un instant *t* que le système ait subi une attaque couronnée de succès et soit encore dans un état où il est susceptible d'être attaqué : celui se traduit par la présence d'un jeton dans la place *Compromis*. C'est la mesure sur laquelle nous allons focaliser nos expériences dans la partie 5 de ce chapitre.

La mesure PC(t) évalue la probabilité à un instant *t* que le système ait subi une attaque couronnée de succès : cela se traduit par la présence présente ou passée d'un jeton dans la place *Compromis*. Cette mesure nécessite donc, afin d'être calculée facilement, l'ajout d'une place supplémentaire, que nous nommerons *C1*, à notre modèle qui contiendra un jeton dès que la place *Compromis* en contiendra un. Ceci sera fait à l'aide d'une transition instantanée et d'une porte d'entrée contenant une pré-condition de test sur le marquage de la place *Compromis*. Une fois le jeton dans la place *C1*, celui-ci ne sera plus déplacé. Ainsi, il sera facile d'évaluer la valeur de PC(t). Nous reviendrons sur l'évaluation de cette mesure dans les expérimentations du chapitre suivant.

PCNR(t) mesure la probabilité à un instant donné *t* que le système ait subi une attaque couronnée de succès et que le système soit susceptible de subir des dégâts causés par une intrusion. Cela se traduit dans le modèle par la présence d'un jeton dans les places *Compromis* ou *Corrige*. L'évaluation de la mesure se fera donc par la simple somme des probabilité de marquage de la place à l'instant *t*.

Enfin, la mesure PS(t) évalue la probabilité à un instant donné *t* que le système soit sûr vis-à-vis de la vulnérabilité. Dans le modèle, cette propriété se traduit par la présence d'un jeton dans la place *Sur*. Le simple contrôle de la probabilité d'un jeton dans cette place suffit donc à évaluer PS(t).

Pour procéder à la validation de notre modèle, nous allons procéder à une première série de simulations en nous concentrant sur l'une de ces quatre mesures : nous évaluons la probabilité que le système soit dans l'état compromis – PPC(t).

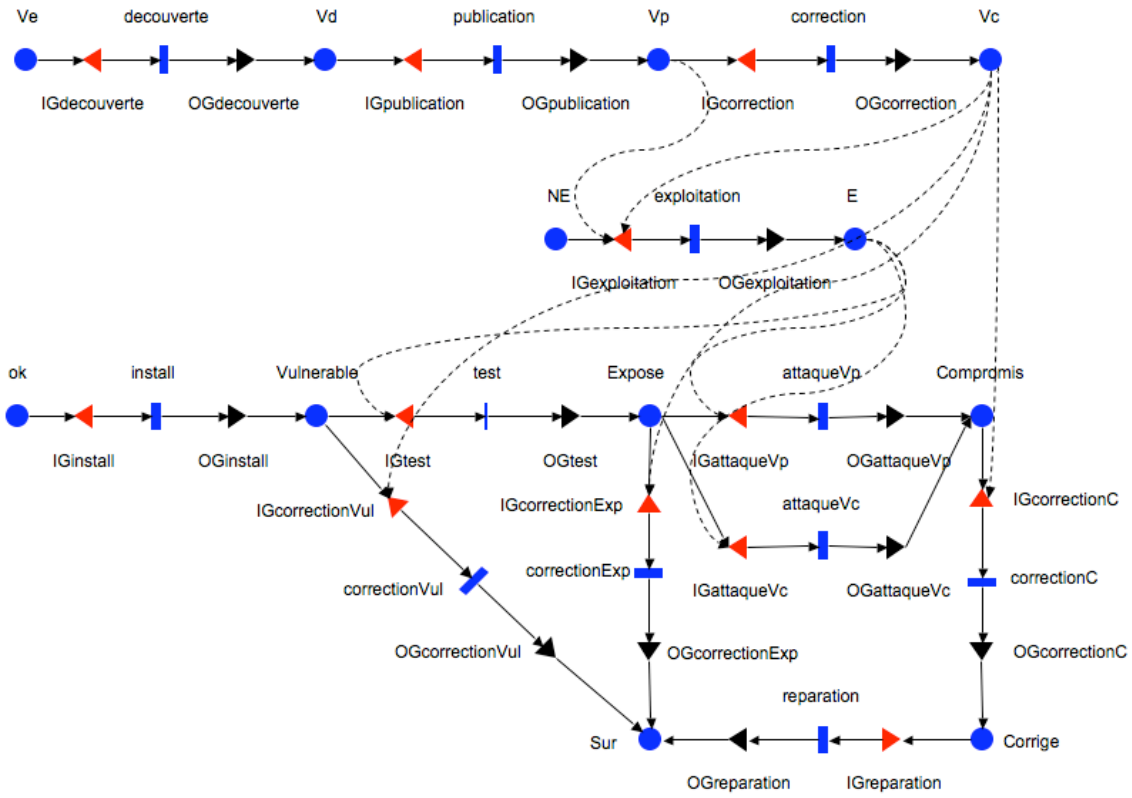


Figure 3.9 : Modèle SAN considérant le scénario de découverte non malveillante

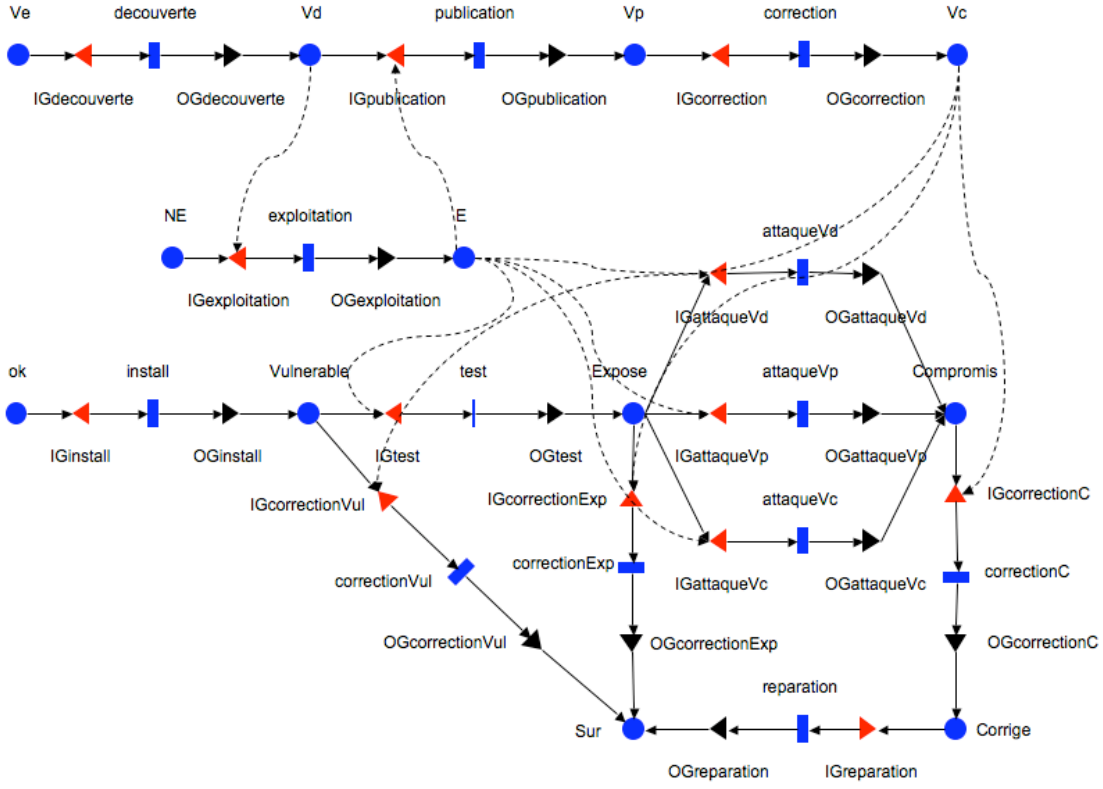


Figure 3.10 : Modèle SAN considérant le scénario de découverte malveillante

5 Validation et résolution

Pour résoudre notre modèle, nous adoptons une démarche de résolution par simulation. En effet, une résolution analytique n'est pas envisageable dans notre contexte où les distributions de probabilité associées aux activités risquent de ne pas être seulement exponentielles. Pour résoudre notre modèle par la simulation, nous utilisons un outil, Möbius, qui permet la simulation de SAN en offrant un paramétrage complet [CLA 01, COU 04]. Dans cette partie, nous présentons dans un premier temps le logiciel Möbius et son fonctionnement. Ensuite, nous présentons nos premières expérimentations, puis analysons nos résultats.

5.1 Premières étapes du fonctionnement de l'outil Möbius

L'outil Möbius a été mis au point pour permettre la simulation et la résolution analytique de modèles dans le cadre de l'évaluation de la performance et de la sûreté de fonctionnement. Pour cela, Möbius propose une démarche étape par étape. Dans cette partie, nous nous focalisons sur la démarche de résolution de modèles par la simulation car c'est celle que nous sommes amenés à utiliser.

5.1.1 L'éditeur du modèle

Möbius propose tout d'abord un éditeur de modèle. Celui-ci permet de faire des réseaux de Pétri, des arbres d'attaques, mais également des SAN. Cet éditeur permet d'éditer facilement les modèles mais également de les compiler pour vérifier la bonne utilisation des éléments du modèle et leur bon paramétrage.

Dans le cas de l'utilisation des SAN, l'éditeur permet également la composition de modèles SAN, appelés modèles atomiques, en proposant les procédés de réplication et de jointure de sous-modèles (eux-mêmes modèles atomiques ou modèles composés). Comme les modèles atomiques, ils sont compilés pour mettre en évidence une mauvaise utilisation des opérateurs ou un manque de paramétrage.

Une fois les modèles édités et compilés, il est possible de poursuivre l'analyse de deux manières distinctes :

- La première est la résolution analytique : cette méthode est possible si le modèle produit peut être traduit facilement en chaîne de Markov, c'est-à-dire si toutes les activités considérées sont associées à une distribution exponentielle.
- La seconde est la résolution par la simulation : elle intervient lorsqu'il est impossible de modéliser toutes les activités par des distributions de probabilité exponentielle.

Pour les raisons que nous avons déjà évoquées dans la partie précédente, nous choisissons la résolution par simulation. En effet, il apparaît peu probable que tous les événements que nous considérons soient caractérisés par des distributions de probabilité exponentielles. Cette hypothèse nous semble trop forte.

5.1.2 Le processus de mesure

Une fois le modèle conçu et compilé, il faut spécifier les mesures que nous allons évaluer. Ces mesures sont spécifiées dans le *Reward Model*. Deux types de mesures sont possibles à partir de deux types de contrôle :

- Le contrôle de la présence de jetons dans une place ;

- Le contrôle de la transition d'une activité.

Notre objectif étant de connaître la probabilité d'être dans l'état compromis, nous définissons notre mesure grâce au premier type de contrôle pour tester la présence d'un jeton dans la place *Compromis*. Ainsi, nous définissons une fonction très simple qui renvoie la valeur 1 lorsque le jeton est détecté dans la place *Compromis*. Il est possible d'effectuer cette mesure plusieurs fois par simulation à des instants donnés. Ces instants seront décidés et paramétrés lorsque nous analyserons les données quantitatives de la vulnérabilité choisie.

La suite de la démarche d'analyse du modèle par l'outil Möbius traite du paramétrage que nous n'avons pas encore présenté. Ainsi, dans le paragraphe suivant, nous présentons la suite de notre démarche de simulation. A chaque étape, nous présentons nos choix et nos décisions, ainsi que les étapes dans l'utilisation de l'outil Möbius.

5.2 Premières simulations

Notre prochaine étape est la simulation de notre modèle. En effet, avant d'aller plus loin, il nous semble important de vérifier le bien fondé de notre démarche et de nous assurer que les facteurs environnementaux que nous avons identifiés constituent une caractérisation correcte de l'environnement : 1) leur influence est-elle significative ? 2) avons-nous oublié un facteur important ? Une première validation nous semble donc indispensable. Pour cela, il est nécessaire de paramétrer notre modèle à l'aide de données réelles. Notre démarche doit donc commencer par la sélection d'une vulnérabilité.

5.2.1 Choix de la vulnérabilité

Pour effectuer nos premières simulations, nous avons besoin d'une vulnérabilité existante, dont les données sont relativement connues. Nous avons besoin de données sur trois critères :

- Le cycle de vie de la vulnérabilité ;
- L'apparition du kit d'exploitation ;
- L'impact en terme de nombre d'intrusions.

L'existence de la vulnérabilité à l'origine du ver Slammer a produit des effets suffisamment remarquables pour avoir suscité une étude plus approfondie. En effet, il est tout de même relativement difficile de pouvoir obtenir des données quantitatives sur une vulnérabilité et les dégâts que son exploitation a engendrés : les administrateurs des systèmes compromis sont majoritairement peu enclins à dévoiler que leur système a été victime d'une attaque réalisée avec succès. Il est donc très difficile d'obtenir des données. De plus, il faut noter que les données, même récoltées, ne sont pas toujours accessibles. En effet, la collecte et la diffusion de données restent problématiques.

Par conséquent, le choix de cette vulnérabilité semble judicieux car les nombreuses études sur cette épidémie semble être une source d'information suffisamment importante.

Cette vulnérabilité est une vulnérabilité des logiciels *Microsoft SQL Server 2000* et *Microsoft Desktop Engine 2000*. Elle permet un débordement de mémoire tampon. Cette vulnérabilité exploitée, il est possible d'exécuter du code malveillant. Rappelons que cette vulnérabilité a été publiée le 25 juillet 2002. Le correctif était disponible le même jour. Un bulletin de sécurité a été publié par Microsoft [MIC 02].

Le ver Slammer exploitant cette vulnérabilité est apparue 185 jours après la publication de la vulnérabilité, soit le 25 janvier 2003. La toute petite taille du ver (376 octets) et le code malveillant qu'il exécute sont à l'origine de son importante propagation.

Cependant, malgré le coté positif de la présence de données, les différentes sources d'information sont plus ou moins précises et la pertinence des données relatées s'en trouve altérée. Dans le paragraphe suivant, nous décrivons quels paramètres nous avons choisis pour la simulation de notre modèle.

5.2.2 Paramètres du modèle

Dans cette partie, nous explicitons les paramètres adoptés pour notre première simulation : nous examinons une à une les trois parties de notre modèle tel que nous l'avons décomposé plus tôt dans ce chapitre : 1) le cycle de vie de la vulnérabilité, 2) le comportement des attaquants et enfin 3) le comportement de l'administrateur et l'état du système. Il est cependant à préciser que nous choisissons le modèle du scénario de découverte d'origine non malveillante correspondant aux données de la vulnérabilité exploitée par le ver Slammer.

5.2.2.1 Paramétrage du cycle de vie de la vulnérabilité

Nous étudions tout d'abord la modélisation du premier facteur environnemental : le cycle de vie de la vulnérabilité. Comme nous connaissons le cycle de vie de la vulnérabilité que nous considérons, nous modélisons l'événement de publication du correctif par une distribution exponentielle avec un taux extrêmement élevé – avec un taux de 1000 jours^{-1} – car la publication de la vulnérabilité et la publication du correctif ont eu lieu en même temps [MOO 03]. Nous aurions pu modéliser cet événement par une activité instantanée. Cependant, nous préférons garder la transition temporisée présente dans le modèle générique. Les taux de découverte et de publication étant bien inférieur, nous leur donnons une valeur de 100 jours^{-1}.

5.2.2.2 Paramétrage du comportement des attaquants

Le second facteur environnemental est le comportement de la population d'attaquants. Dans [FRE 06], les auteurs montrent que la création d'un kit d'exploitation peut avoir lieu selon une distribution de probabilité de Pareto. Cependant, comme nous connaissons l'exacte parution du vers Slammer, nous avons modélisé la création du vers par une distribution déterministe. Cela signifie que l'activité *exploitation* sera franchie 185 jours après la publication de la vulnérabilité. Les phénomènes d'attaque sont modélisés par des distributions exponentielles, comme ils l'ont été dans [ORT 99, KUH 07]. Ces distributions ont, en outre, été choisies pour leur décroissance rapide et non pas pour leur propriété de modélisation d'un phénomène sans mémoire. Les données présentes dans [MOO 03] ont servi de base pour calculer le taux d'attaque. Nous nous sommes basés sur les données fournies pour calculer un taux à partir de l'estimation de 75000 infections dans les dix premières minutes de propagation (certaines sources estimant le ver plus ou moins virulent, nous avons choisi l'estimation la plus fréquemment avancée). Nous obtenons un taux de 23,4 jours^{-1}que nous appliquons aux deux activités *attaqueVp* et *attaqueVc*. Néanmoins, le modèle SAN que nous avons développé dans notre approche peut être paramsétré en utilisant d'autres types de distributions.

5.2.2.3 Paramétrage de l'état du système et du comportement de l'administrateur

La dernière partie du modèle combine l'état du système et le comportement de l'administrateur. Nous commençons le paramétrage par le phénomène d'installation du composant vulnérable. Dans le cas de la vulnérabilité du ver Slammer, le composant vulnérable était, dans une quasi totalité des cas, déjà présent sur le système. Nous avons donc choisi de modéliser cet événement d'installation par une distribution exponentielle avec un taux très important.

L'action principale de l'administrateur est l'application du correctif de la vulnérabilité. Comme nous l'avons vu précédemment, il peut intervenir dans trois circonstances différentes. Néanmoins, nous choisissons de modéliser son comportement à chaque fois avec une distribution normale, dont nous adaptons les paramètres en fonction de l'administrateur.

Mais l'administrateur a également besoin de réparer le système lorsque celui-ci a été compromis. Ainsi, il y a une phase de réparation nécessaire que nous caractérisons avec une distribution exponentielle. Les conséquences du ver Slammer pour le système n'étant pas catastrophiques, nous avons compté un délai moyen de réparation d'une journée.

Enfin, nous avons mis un taux d'installation très élevé car le composant vulnérable était déjà installé dans la majorité des cas. Néanmoins, pour ne pas enlever la généricité du modèle, nous avons choisi de ne pas changer notre transition temporisée en transition instantanée.

Dans la partie suivante, nous détaillons nos simulations et spécifions les paramètres quantitatifs appliqués à notre modèle selon nos expérimentations.

5.2.3 Préparation des simulations

Cette première série d'expériences a pour but de valider notre modèle d'un point de vue logique et de vérifier la pertinence des événements environnementaux considérés. Nous voulons, avant d'aller plus loin dans notre démarche, valider plusieurs points de notre approche.

Le premier est la bonne identification des facteurs environnementaux : nous voulons nous assurer que les trois facteurs que nous avons identifiés ont bien une influence significative. Le second point est que nous avons correctement analysé leur évolution et leurs dépendances et interdépendances. Cela signifie que nous pouvons poursuivre notre travail en conservant ces premiers travaux comme base.

Nous avons mis au point trente expériences en nous concentrant sur le facteur environnemental le plus local : le comportement de l'administrateur. Nous nous sommes concentrés sur le temps que ce dernier mettait à appliquer le correctif de la vulnérabilité. Nous avons donc, d'après le modèle, pris en considération trois temps d'application du correctif selon les trois phases possibles :

- Lorsque le système est vulnérable : la vulnérabilité est présente sur le système mais il n'existe pas de kit d'exploitation pour la vulnérabilité.
- Lorsque le système est exposé : la vulnérabilité est présente sur le système et il existe un kit d'exploitation pour mener à bien une attaque. On peut considérer que c'est l'état où le système est en danger.
- Lorsque le système est compromis.

Comme nous avons modélisé le phénomène d'application du correctif par l'administrateur par une distribution de probabilité normale, nous associons donc à chaque expérience trois temps de réaction moyens selon l'état du système. Nous définissions le rapport entre les trois temps de réaction selon la formule ci-dessous :

$$\alpha_{Vul} = 2\ \alpha_{Exp} = 100\ \alpha_{C}$$

La formule détaille les rapports entre les temps moyens de réaction que nous avons attribués. α_{Vul} représente le temps moyen d'application du correctif lorsque le système est dans l'état vulnérable. α_{Exp} représente le temps moyen d'application du correctif lorsque le système est dans l'état exposé. α_{C} représente le temps moyen d'application du correctif lorsque le système est dans l'état compromis. La formule indique que l'administrateur est deux fois plus réactif lorsque le système est exposé plutôt que vulnérable : en effet, il peut avoir eu vent de l'existence du kit d'exploitation et se montrer plus attentif. Lorsque le système est compromis, l'administrateur est forcément beaucoup plus réactif et a à cœur de protéger son système dans les plus brefs délais. Ainsi, l'administrateur réagira 100 fois plus rapidement dans le cas où le système est compromis que lorsqu'il est vulnérable. Ainsi, les trente expériences ont des temps moyens d'application du correctif différents, résumés dans le tableau 3.3. Le second paramètre de la distribution normale, la variance a été fixée à 0,5 pour les trois comportement, pour avoir une dispersion constante et peu importante. En effet, nous voulions une distribution de probabilité qui ne couvre pas un trop grand intervalles de comportements. Les autres paramètres appliqués aux distributions de probabilité modélisant les différents événements sont détaillés dans le tableau 3.4.

N° d'expérience	α_{Vul}	α_{Exp}	α_{C}	N° d'expérience	α_{Vul}	α_{Exp}	α_{C}
1	10	5	0.1	16	160	80	1.6
2	20	10	0.2	17	170	85	1.7
3	30	15	0.3	18	180	90	1.8
4	40	20	0.4	19	190	95	1.9
5	50	25	0.5	20	200	100	2.0
6	60	30	0.6	21	210	105	2.1
7	70	35	0.7	22	220	110	2.2
8	80	40	0.8	23	230	115	2.3
9	90	45	0.9	24	240	120	2.4
10	100	50	1.0	25	250	125	2.5
11	110	55	1.1	26	260	130	2.6
12	120	60	1.2	27	270	135	2.7
13	130	65	1.3	28	280	140	2.8
14	140	70	1.4	29	290	145	2.9
15	150	75	1.5	30	300	150	3.0

Tableau 3.3 : Paramétrage du temps moyen en jours d'application du correctif pour les 30 expériences

Activité	Distribution	Paramètre	Valeur
Decouverte	Exponentielle	Taux	100 jours^{-1}
Publication	Exponentielle	Taux	100 jours^{-1}
Correction	Exponentielle	Taux	1000 jours^{-1}
Exploitation	Déterministe	Instant	185 jours
Install	Exponentielle	Taux	1000 jours^{-1}
correctionVul	Normale	Moyenne (α_{Vul})	10 ⇨ 300 jours
		Variance	0.5 jours²
correctionExp	Normale	Moyenne (α_{Exp})	5 ⇨ 150 jours
		Variance	0.5 jours²
correctionC	Normale	Moyenne (α_C)	0.1 ⇨ 3 jours
		Variance	0,5 jours²
attaqueVp	Exponentielle	Taux	23.4 jours^{-1}
attaqueVc	Exponentielle	Taux	23.4 jours^{-1}
Reparation	Exponentielle	Taux	1 jour^{-1}

Tableau 3.4 : Résumé du paramétrage du modèle appliqué à la vulnérabilité du ver Slammer

Dans le logiciel Möbius, le paramétrage du modèle est contenu dans une "étude" (*Study* en anglais) associée au modèle qui contient les valeurs quantitatives des paramètres associés aux distributions choisies dans le modèle. Cette étude contient ainsi les paramètres détaillés dans ce paragraphe. Les différentes valeurs attribuées aux trois paramètres α_{Vul}, α_{Exp} et α_C sont calculées à partir d'une valeur de départ et d'un incrément.

Cette étude va permettre de générer le modèle de simulation qui regroupe l'ensemble des expériences qui seront exécutées. En fait, ce modèle de simulation contient autant d'expériences qu'il y a de combinaisons de paramètres possibles. L'outil Möbius permet de sélectionner les expériences à exécuter qui correspondent à nos choix de comportements pour l'administrateur : les expériences correspondant aux paramètres décrits dans le tableau 3.3.

Chacune de ces expériences a été simulée un minimum de 10 000 fois pour obtenir des résultats pertinents. Ces résultats sont présentés dans la partie suivante.

5.2.4 Résultats et analyses

Après avoir exécuté les trente expériences relatives aux trente comportements modélisés pour l'administrateur, nous avons récupéré et analysé les résultats édités par le simulateur. Celui-ci produit un fichier par expérience, contenant les résultats des mesures effectuées, ainsi que le rappel des paramètres utilisés. Pour cette série de simulations, nous nous sommes particulièrement intéressés aux temps moyens de compromission du système en considérant un comportement particulier de l'administrateur. Pour chaque expérience, nous avons mesuré quelle était la probabilité pour que le système soit dans l'état compromis à des instants précis de la simulation. Nous avons effectué cette mesure pour chaque jour simulé entre le premier et le 500^{e} jour.

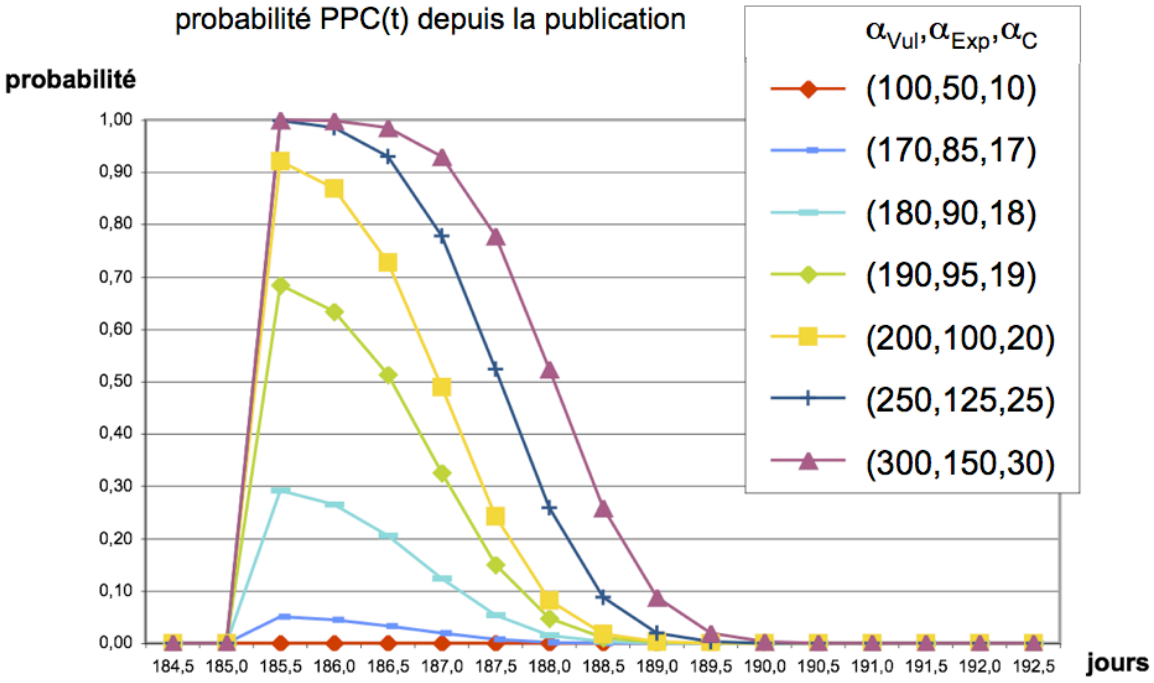


Figure 3.11 : Résultats des simulations mesurant la probabilité PPC(t) de compromission du système par l'exploitation de la vulnérabilité du ver Slammer

La figure 3.11 montre les résultats obtenus par nos simulations. Les mesures sont présentées à partir du 185e jour, jour d'apparition du ver exploitant la vulnérabilité jusqu'au 192e jour. La figure 3.11 ne représente pas également tous les profils de comportement de l'administrateur : nous n'avons gardé que quelques-uns pour illustrer plus clairement les résultats.

Notre première observation concerne l'allure générale de la courbe. Celle-ci est similaire à ce qui a pu être observé dans la réalité. En effet, on sait que cette vulnérabilité était présente sur la plupart des systèmes utilisant le système d'exploitation vulnérable. Cela signifie donc que le correctif de la vulnérabilité avait été très peu appliqué. Nous sommes partis sur un taux de compromission de 23,4 systèmes par jour, ce qui est vraisemblablement sous-estimé : en effet, l'intégralité des victimes n'a sûrement pas été recensée. Cependant, on peut voir sur la courbe présentée sur la figure 3.11 qu'il suffit d'une durée d'application moyenne du correctif soit supérieure à 200 jours en phase de système vulnérable pour que la probabilité d'infection soit supérieure à 90% dans la première demie journée.

La seconde observation concerne les répercutions du comportement de l'administrateur. Nous avons choisi de nous concentrer sur le comportement de l'administrateur pour mettre en avant l'influence de ce facteur. Les différentes courbes de la figure 3.11 mettent en évidence cette influence : un administrateur ayant le comportement correspondant au triplet de paramètres (100,50,1) ne fait encourir aucun risque a son système. A l'extrême inverse, l'administrateur ayant un comportement caractérisé par le triplet de paramètres (300,150,30) augmentera de manière très significative la probabilité pour son système d'être compromis.

Les résultats obtenus par nos premières simulations semblent, à première vue, relativement évidents : plus l'administrateur est rigoureux et conscient des objectifs de

sécurité, moins le système a de risque d'être compromis. Cependant, le rôle de ces simulations était de valider notre approche et non de fournir de nouveaux résultats. De plus, ces simulations peuvent être utiles dans un autre cadre, celui d'un rapport entre coût et maintien du niveau de sécurité, comme cela est discuté dans le paragraphe suivant.

5.2.5 Rapport entre coût et maintien du niveau de sécurité

Nous avons vu dans la partie de présentation et d'analyse des résultats l'influence très significative du comportement de l'administrateur. Ainsi, à partir du profil d'un administrateur, traduit par les temps moyens d'application du correctif, il est donc possible d'évaluer une probabilité de compromission du système par l'exploitation d'une vulnérabilité. Cette mesure correspond au premier point de vue que nous avions énoncé, à savoir l'évaluation des risques en considérant une situation donnée.

En considérant le point de vue inverse, il est possible d'estimer l'investissement nécessaire de l'administrateur pour maintenir un niveau de sécurité. Ainsi, il est possible, à partir d'un niveau de risque que l'administrateur est prêt à assumer, c'est à dire une valeur de la probabilité de compromission du système, de quantifier l'investissement minimal de l'administrateur. Cette démarche correspond au second point de vue que nous avions énoncé : l'évaluation d'une distance à objectif. En effet, en ayant caractérisé son comportement actuel et celui sont lequel le niveau de sécurité exigé est assuré, l'administrateur évalue la distance entre son comportement actuel et celui qu'il doit avoir.

Illustrons cette idée par un exemple. Un administrateur administre un parc de systèmes fonctionnant avec le système d'exploitation Windows. La société Microsoft a pour habitude de diffuser ces correctifs sous forme de mise à jour téléchargeable un mardi par mois (appelé "Patch Tuesday"). L'administrateur applique donc tous les mois les correctifs disponibles. Cependant, certains correctifs sont diffusés hors de ce cycle mensuel en cas de plus grand danger, lorsqu'un kit d'exploitation est disponible. Il est donc possible, en caractérisant ce comportement de déterminer si l'attitude va suffire à assurer le niveau de sécurité souhaité. Et dans le cas contraire, il sera alors possible d'évaluer à quelle fréquence il est nécessaire de surveiller les correctifs des vulnérabilités dangereuses – possédant un kit d'exploitation disponible – pour maintenir la sécurité du système.

Le bilan de notre démarche est plutôt positif. Notre modélisation permet de produire des mesures pour la sécurité qui semblent réalistes : l'influence des facteurs environnementaux est donc effective et correctement caractérisée. Cependant, nous nous sommes volontairement limités à modéliser le processus de compromission du système en considérant une seule vulnérabilité. Dans la partie suivante, nous présentons les premiers travaux d'extension de notre modèle pour prendre en considération plusieurs vulnérabilités.

6 Conclusion

Dans ce chapitre, nous avons présenté en détail les modèles que nous avons mis au point en vue de produire des mesures de la sécurité. Nous avons tout d'abord détaillé comment il était possible de modéliser chaque facteur environnemental ainsi que l'état du système. Nous avons ensuite réuni ces premières modélisations et incorporé les interdépendances que nous connaissions pour créer deux modèles de processus de compromission du système par l'exploitation d'une vulnérabilité. Chacun de ces modèles

correspond à un scénario établi depuis l'origine de la découverte de la vulnérabilité considérée.

Avant de poursuivre plus loin nos travaux, il nous est apparu important de valider notre approche. Nous avons donc utilisé l'environnement de modélisation et de simulation Möbius pour effectuer une série de simulations à partir des données quantifiées par les études menées sur la vulnérabilité exploitée par le ver Slammer. Ces premières simulations ont donné des résultats satisfaisants, nous autorisant à approfondir notre approche.

Cependant, les événements du cycle de vie et de la mise au point du kit d'exploitation nécessite une plus grande attention. Le prochain chapitre présente notre étude pour la caractérisation de ces événements à partir de données réelles, ainsi que les expérimentations menées pour l'évaluation des mesures définies.

Chapitre 4 : Caractérisation des événements et expérimentations

Nous avons présenté l'élaboration de notre modélisation permettant de fournir les mesures quantitatives pour la sécurité. Nous avons fait cette modélisation en deux temps. Dans une première étape, nous avons mis au point un modèle du processus de compromission d'un système d'information par l'exploitation d'une vulnérabilité. La seconde étape a été la prise en compte de plusieurs modèles pour mener une exécution en parallèle. Nous avons détaillé, dans le chapitre précédent, l'élaboration de notre modèle. Nous avons également évoqué le problème de la caractérisation de certains événements par des distributions de probabilité. Dans ce chapitre, nous nous penchons sur cette problématique avant de présenter les résultats des simulations paramétrées.

1 Caractérisation des événements

Il est important, pour que notre modèle puisse produire des résultats utilisables, de caractériser les événements par des distributions de probabilité correspondant au comportement réel. Dans cette partie, nous présentons la méthodologie et les résultats pour la caractérisation des événements de notre modèle.

1.1 Caractérisation des événements du cycle de vie et de la mise au point du kit d'exploitation

Les premiers événements que nous caractérisons sont les événements du cycle de vie de la vulnérabilité et la mise au point du kit d'exploitation. Dans cette section nous présentons d'abord quelques travaux sur la caractérisation de tels événements avant de présenter notre démarche en détail.

1.1.1 Approches pour la caractérisation des événements du cycle de vie

Il existe plusieurs approches pour la caractérisation des vulnérabilités par leur cycle de vie. Dans [ARO 04], les auteurs étudient l'impact de la publication de la vulnérabilité sur le processus d'attaque. L'ensemble de vulnérabilités utilisé pour mener cette étude compte 308 vulnérabilités. Les statistiques obtenues permettent de paramétrer un modèle économique évaluant l'évolution du nombre d'attaques prévues par machine et par jour. Mais il est nécessaire de donner comme paramètre du modèle les dates de publication de la vulnérabilité et du correctif de la vulnérabilité.

Dans [JUM 08], les auteurs étudient 240 vulnérabilités en ayant rassemblé des informations sur leur cycle de vie. Le cycle de vie de chaque vulnérabilité peut être associé à une des catégories énoncées par les auteurs, comme par exemple *Zero-Day attack*, ou encore *Potential for attack*, qui regroupe les vulnérabilités possédant un kit d'exploitation

disponible mais pas encore de correctif. A partir des événements du cycle de vie considérés et des caractéristiques de la vulnérabilité, les auteurs ont établi une mesure à partir de huit facteurs concernant l'état de la vulnérabilité, comme par exemple l'âge de la vulnérabilité (jeune, normale ou vieille), la fenêtre de risque (grande, moyenne ou petite), ou encore l'existence d'un kit d'exploitation (existant, non existant ou inconnu). Chaque axe est estimé de façon qualitative selon une échelle ordonnée comportant trois états possibles. A partir de ces huit estimations, une mesure globale est déduite d'une somme pondérée puis normalisée des estimations qualitatives. Ces mesures sont ensuite analysées à partir des catégories de vulnérabilités établies. Cette étude met en évidence l'absence, chez Microsoft, de correctif pour beaucoup de vulnérabilités.

Dans [FRE 06], les auteurs basent leur étude sur un ensemble de 14326 vulnérabilités, issues de plusieurs bases de données. Les analyses menées sur ces données se concentrent sur le phénomène de publication de la vulnérabilité, c'est-à-dire que les autres événements étudiés, l'apparition du kit d'exploitation et la publication du correctif de la vulnérabilité, le sont par rapport à la date de publication de la vulnérabilité. Ce travail semble être très intéressant pour nous, mais, malheureusement, ne considère pas la découverte de la vulnérabilité, très important dans notre approche. Il s'avère donc impossible pour nous, à partir de cette étude, de :

- quantifier la publication de la vulnérabilité en fonction de l'événement de découverte de la vulnérabilité ;
- de caractériser l'apparition du kit d'exploitation à partir de ce même événement de découverte de la vulnérabilité.

Il nous faut donc caractériser par nous même les phénomènes liés au cycle de vie de la vulnérabilité et à l'apparition du kit d'exploitation. Pour cela, nous allons d'abord nous concentrer sur l'étude des bases de données de vulnérabilités existantes sur lesquelles nous pourrons nous appuyer.

1.1.2 Caractérisation des événements par l'analyse d'une base de données

Le travail de caractérisation des événements se décompose en deux étapes. La première consiste en l'analyse des bases de données existantes afin d'obtenir un jeu de données aussi adapté que possible à l'étude que nous voulons mener. Une fois les données sélectionnées, la seconde étape consiste en l'analyse de ces données en vue de caractériser les événements de notre modèle. Pour cela, nous avons d'abord établi une méthodologie d'analyse avant de l'appliquer à nos données pour produire nos résultats.

1.1.2.1 Sélection de la base de données

Pour caractériser de manière quantitative les événements de notre modélisation, il est nécessaire de se baser sur un ensemble de données aussi complet que possible afin d'obtenir des résultats pertinents. Le plus complet des ensembles de vulnérabilités est l'union des ensembles de vulnérabilités contenus dans les bases de données existantes. Malheureusement, il est très difficile d'obtenir cet ensemble de vulnérabilités. CVE (*Common Vulnerability Exposure*) propose un numéro de référence de la vulnérabilité qui pourrait être un facteur commun. Il est présent pour chaque description de vulnérabilités dans la base NVD (*National Vulnerablity Database*) [NVD]. Mais malheureusement, ce référencement n'est pas systématique dans les autres bases de données, ce qui rend difficile la création d'une base globale.

Nous allons donc analyser les bases de données existantes pour déterminer si une base en particulier peut nous servir à caractériser les événements de notre modélisation et

laquelle choisir. En effet, pour choisir la base de données susceptible de convenir à notre analyse, nous devons étudier plus en détail quelles informations sont stockées dans chacune d'elle. Nous avons concentré notre étude sur les quatre bases de données de vulnérabilités les plus utilisées :

- NVD : c'est une base de données de vulnérabilités gérée par le NIST (*National Institute of Standards and Technologies*), aux Etats Unis. Cette base de données a été mise en place en 1999. Les vulnérabilités sont classées dans 23 catégories, qui sont pour la plupart, des catégories CWE (*Common Weaknesses Enumeration*), définies par le NIST. Ces catégories de vulnérabilités ont été définies suite à l'analyse de vulnérabilité et leur définition grâce au langage de spécifications associé à CWE. L'axe de classification est principalement le type d'attaque utilisé pour exploiter la vulnérabilité. Pour chaque vulnérabilité, la date de publication est généralement indiquée, ainsi que la dernière date de modification des informations relative à la vulnérabilité, l'existence d'un correctif ou d'une solution (dans le cas d'une vulnérabilité de configuration), la liste des composants vulnérables, et une évaluation de la gravité de l'impact de la vulnérabilité basée sur le système de mesure CCVS [MEI 07].
- Security Focus [SECa] : la base de données de Security Focus gérée depuis 2002 par *Symantec Corporation*. Elle contient environ 35000 vulnérabilités, enregistrées depuis octobre 1998. Les vulnérabilités sont classées dans 11 classes différentes. Les renseignements fournis sont la date de publication de la vulnérabilité, la date de dernière modification des informations relatives a la vulnérabilité, l'existence d'un correctif ou d'une solution et la liste des composants vulnérables.
- OSVDB [OSV] : cette base de données est une base *open source* créée en 2002 par les participants de la conférence *Black Hat* et ouverte en libre accès au public depuis 2004. Elle contient plus de 52000 vulnérabilités recensées depuis 1998. Les vulnérabilités sont classées en 9 classes, mais une vulnérabilité peut être associée à aucune ou plusieurs classes. Les renseignements fournis sont les dates de découverte de la vulnérabilité, de publication de la vulnérabilité, de publication du correctif et d'apparition du kit d'exploitation si ce dernier existe. La liste des composants vulnérables est également fournie.
- Secunia [SECb] : Secunia est une société de service privée qui travaille sur l'analyse des vulnérabilités et la protection des sociétés dans le domaine de la sécurité. Elle gère une base de données de plus de 30000 vulnérabilités en libre accès au public depuis 2002 et propose une classification des vulnérabilités en 12 classes, mais renseigne aussi la date de publication de la vulnérabilité, l'existence d'un correctif ou d'une solution et les renseignements pour appliquer celui-ci, la liste des composants vulnérables et indique la gravité de l'impact de l'exploitation de la vulnérabilité par la mesure CCVS.

Quelque soit la base de données, l'axe de classification est toujours le type d'attaque utilisé pour exploiter la vulnérabilité.

Les caractéristiques de chaque base de données de vulnérabilités sont résumées dans le tableau 4.1. Il apparaît que la base de données OSVDB correspond le mieux à nos exigences. Elle offre un maximum de renseignements temporels, tant sur les événements du cycle de vie de la vulnérabilité que sur l'événement d'apparition du kit d'exploitation.

Base de données	NVD	Security Focus	OSVDB	Secunia
Date de découverte	Non	non	oui	non
Date de publication	Oui	oui	oui	oui
Date de publication du correctif	Non	non	oui	non
Date d'apparition du kit d'exploitation	Non	non	oui	non
Date de dernière modification	Oui	oui	oui	non
Existence d'une solution ou correction	Oui	oui	oui	oui
Catégorie de la vulnérabilité	Oui	oui	oui	oui
Sévérité évaluée par CCVS	Oui	non	non	oui
Liste des composants vulnérables	Oui	oui	oui	oui

Tableau 4.1 : Comparaison des bases de données de vulnérabilités

Vulnérabilités	52000
Vulnérabilités avec date de découverte	3961
Vulnérabilités avec date de publication	51099
Vulnérabilités avec date de publication du correctif	1151
Vulnérabilités avec date d'apparition du kit d'exploitation	17857

Tableau 4.2 : l'ensemble de vulnérabilités étudié

Nous choisissons cette base de données pour travailler. Elle nous permet d'obtenir une base de 52000 vulnérabilités, enregistrées entre décembre 1998 et janvier 2009. Nous enregistrons, pour chaque vulnérabilité, son identifiant unique délivré par les gestionnaires de la base de données, la date de découverte de la vulnérabilité, la date de publication, la date de publication du correctif, la date d'apparition du kit d'exploitation et les catégories de vulnérabilités associées. Malheureusement, l'ensemble de ces données n'est pas renseigné pour chaque vulnérabilité, mais cela reste un défaut de chaque base de données rencontrée. Nous résumons dans le tableau 4.2 les données que nous avons récupérées.

Dans la section suivante, nous menons une première analyse de ces données. Par la suite, nous présentons la méthodologie d'analyse des données sur les vulnérabilités. Enfin, nous présentons les résultats obtenus.

1.1.2.2 Analyses préliminaires

Avant d'analyser en profondeur les données qui nous sont fournies, nous commençons par une analyse préliminaire. Comme cela est résumé dans le tableau 4.2, nous avons à disposition un nombre relativement important de données pour chaque événement du cycle de vie de la vulnérabilité mais également pour l'événement d'apparition du kit d'exploitation. Cependant, il est important de remarquer que le nombre de vulnérabilités renseignées pour chaque événement varie et ces variations peuvent être en elles-mêmes des sources d'information.

Evénements	Total	(%)	Découverte	(%)	Publication	(%)	Correction	(%)	Kit d'exploit.	(%)
Découverte	3961	7,62	3961	100,00	3926	7,68	148	12,86	2131	11,93
Publication	51099	98,27	3926	99,12	51099	100,00	871	75,67	17857	100,00
Correction	1151	2,21	148	3,74	871	1,71	1151	100,00	290	1,62
Kit d'exploit.	17857	34,34	2131	53,80	17857	34,95	290	25,20	17857	100,00

Tableau 4.3 : Récapitulatif des données

Le tableau 4.3 présente le nombre de chaque ensemble de vulnérabilités pour lequel deux événements sont renseignés. Les cases colorées représentent les nombres de vulnérabilités en ne considérant qu'un seul événement. Dans les cases blanches, sont recensés les nombres de vulnérabilités pour lesquelles deux événements sont renseignés. La proportion indiquée à coté correspond à la proportion du nombre de vulnérabilités possédant les deux informations par rapport au nombre de vulnérabilités possédant l'information correspondant à l'événement de la colonne.

Pour clarifier le tableau, prenons un exemple simple. Considérons les vulnérabilités ayant une date d'apparition du kit d'exploitation, soit la dernière colonne de notre tableau. La dernière ligne représente le nombre total de cette catégorie de vulnérabilités, spécifié dans la case colorée (on retrouve cette information dans la dernière case de la première colonne, qui indique également la proportion de cette catégorie de vulnérabilités par rapport au jeu de données total). Les cases précédentes indiquent le nombre de vulnérabilités possédant une date de parution de kit d'exploitation et une autre information relative au cycle de vie (date de découverte, de publication ou de publication du correctif). La proportion en pourcentage est calculée par rapport au nombre de vulnérabilités possédant la caractéristique de la date de parution du kit d'exploitation.

Nous nous intéressons à l'ensemble des vulnérabilités qui sont à la fois renseignées pour la date de découverte et la date de publication. Il y en a 3926, ce qui représente une petite proportion du nombre total de vulnérabilités étudiées (7.62%), mais seulement 99.12% de l'ensemble des vulnérabilités possédant une date de découverte connue. Le cardinal faible de cet ensemble de vulnérabilités peut donc s'expliquer par le cardinal faible de l'ensemble des vulnérabilités possédant une date de découverte connue. Cela s'explique néanmoins de manière très logique puisque cet événement n'est pas un événement du cycle de vie de la vulnérabilité publié officiellement.

Le second ensemble de vulnérabilités auquel nous nous intéressons est l'ensemble des vulnérabilités possédant une date de publication et une date de correctif. Cet ensemble comporte 871 vulnérabilités. Nous nous retrouvons dans le même cas de figure que précédemment : cet ensemble ne représente que 1.71% de l'ensemble des vulnérabilités comportant une date de publication connue alors qu'il représente 75.67% de l'ensemble des vulnérabilités possédant une date de publication du correctif connue. Cependant, il apparaît plus difficile d'expliquer le petit nombre de vulnérabilités possédant une date de publication du correctif. Cette petite proportion peut avoir deux explications :

- Seule une petite proportion des vulnérabilités publiées est corrigée. Cela apparaît comme une explication possible car nous considérons le phénomène de publication de vulnérabilité comme un phénomène global, ne dépendant pas nécessairement du producteur du composant vulnérable. Cette publication pouvant être effectuée par un tiers, celle-ci n'est pas automatiquement suivie de l'apparition d'un correctif. Plusieurs études, comme [JUM 08] évoquent ce phénomène : elles avancent qu'une toute petite proportion des vulnérabilités publiées ont, à terme, un correctif disponible.

- L'ensemble des vulnérabilités que nous analysons dans cette section n'est pas exhaustif et nous ne devons pas oublier que l'absence d'information sur l'existence d'un correctif ne signifie pas que celui-ci n'existe pas, mais que l'information n'a pas été stockée dans la base de données.

Il est impossible d'ignorer l'une ou l'autre de ces deux explications. Cependant, les études citées nous permettent de ne pas négliger l'explication selon laquelle une grande proportion des vulnérabilités publiée ne sera pas corrigée. Nous choisissons donc de considérer notre base de données comme la plus exhaustive possible et de privilégier la première explication. Cette décision est motivée par les études citées déjà existantes.

Enfin, nous examinons les ensembles de vulnérabilités considérant la date d'apparition du kit d'exploitation. Les cardinaux de ces trois ensembles de vulnérabilités apparaissent dans la dernière colonne du tableau 4.3. Il est intéressant de noter que l'ensemble des vulnérabilités possédant une date d'apparition du kit d'exploitation connue ont également toutes une date de publication disponible, ce qui optimise le cardinal de notre ensemble de données. Cet ensemble ne représente cependant qu'environ 35% des vulnérabilités publiées. Il apparaît important de garder en tête cette proportion et de confronter cette donnée avec l'étude plus approfondie que nous comptons mener.

Il est également important de remarquer que la proportion de vulnérabilités possédant un kit d'exploitation et un correctif est très faible. Cette proportion est du même ordre de grandeur que celle des vulnérabilités publiées possédant un correctif.

Cette analyse préliminaire a permis de mettre en évidence une information importance qui est la grande proportion de vulnérabilités ne possédant pas de correctifs. Il faudra donc tenir compte de cette information lors du paramétrage de notre modèle. Dans le paragraphe suivant, nous expliquons notre démarche pour la caractérisation des événements à partir des données récupérées à partir de la base de données de vulnérabilités OSVDB.

1.1.2.3 Méthodologie pour la caractérisation des événements

Une fois les données récupérées depuis la base de données, il est nécessaire d'analyser ces données pour déterminer s'il est possible de trouver des distributions de probabilité susceptibles de caractériser les différents événements du cycle de vie de la vulnérabilité ainsi de la mise au point du kit d'exploitation.

Pour chaque événement que nous voulons caractériser, nous collectons et analysons les données qui nous seront utiles. Pour caractériser un événement depuis un état i à un état j du système, nous sélectionnons les vulnérabilités pour lesquelles les dates t_i et t_j nous sont connues afin d'évaluer la durée t_j-t_i. Ainsi, nous possédons un nouvel ensemble de données composé des durées évaluées entre l'état i et l'état j. Nous avons besoin de classer ces données pour être capable de les analyser et trouver la distribution de probabilité la plus appropriée.

Grouper les données par classes permet d'ignorer les petites variations non significatives pour se focaliser sur la tendance générale. A l'inverse, il est très important de ne pas choisir un nombre de classes trop faible qui pourrait amener à masquer des informations importantes pour l'analyse. Nous inclination naturelle serait de prendre un nombre de classes le plus important possible pour coller au maximum aux données réelles : nous nous basons sur la formule de Sturges [STU 26] afin de déterminer le nombre de classes optimal. Les classes sont composées du même nombre de données.

Une fois ces données traitées, nous avons utilisé l'outil EasyFit [MAT] pour comparer les histogrammes des données obtenus et les comparer aux distributions de

probabilité. Le test de Kolmogorov-Smirnov indique si la distribution de probabilité convient. Ce test est un test d'ajustement permettant de confronter des données à une loi de probabilité théorique. Il teste l'hypothèse selon laquelle ces données ont été engendrées par une loi de probabilité qui admet comme modèle convenable la loi de probabilité théorique. Il est utilisé lorsque la loi théorique est une distribution de probabilité continue. Son principe repose sur l'étude de la différence entre la fonction de répartition empirique des données et la fonction de répartition théorique. Ce test statistique fournit un résultat permettant de guider le choix de rejet ou d'acceptation de l'hypothèse en fonction d'un seuil de confiance. Nous choisissons pour notre étude un seuil de 0,05.

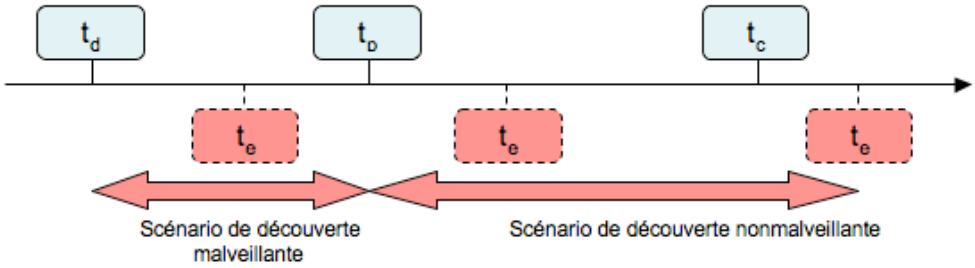
1.1.2.4 Résultats

Nous avons analysé nos données pour évaluer les durées entre événements du cycle de vie de la vulnérabilité en vue de caractériser les événements de publication de la vulnérabilité et de publication de son correctif. A partir de ces données, il est impossible de caractériser l'événement de découverte de la vulnérabilité car il est difficile de connaître la date à partir de laquelle la vulnérabilité existe. En tout cas, cette information de date d'existence de la vulnérabilité n'est recensée dans aucune base, ce qui rend le travail d'évaluation très difficile. Dans cette première partie, nous présentons nos résultats pour la caractérisation des événements de publication de la vulnérabilité et de la publication de son correctif, ainsi que l'événement de mise au point du kit d'exploitation par la population des attaquants.

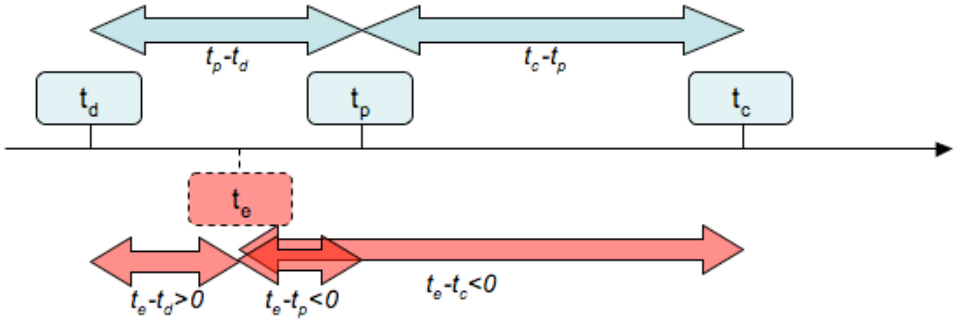
1.1.2.4.1 Etude du phénomène de publication de la vulnérabilité

La publication de la vulnérabilité est un événement qui peut survenir après la découverte d'origine non malveillante de la vulnérabilité, ou encore après l'utilisation du kit d'exploitation par la population d'attaquants. Notre but est d'évaluer l'évolution de la durée entre découverte et publication de la vulnérabilité. Cette analyse a pour but d'aider la caractérisation du phénomène de la publication de la vulnérabilité dans notre modélisation.

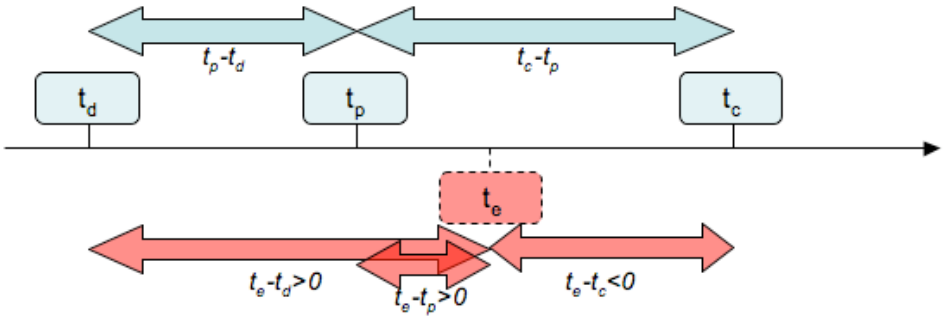
Il est important de tenir compte des deux scénarios que nous avons identifiés à partir de l'origine de la découverte de la vulnérabilité. Nous rappelons les différents cas de figure susceptibles de survenir dans la figure 4.1. Les symboles t_d, t_p, t_c et t_e représentent les dates, respectivement, de découverte, publication, publication du correctif et apparition du kit d'exploitation. Le schéma *a* montre les intervalles de temps durant lesquels l'apparition du kit d'exploitation peut avoir lieu. Chaque cas de figure est ensuite détaillé dans les schémas *b*, *c*, et *d*. Le signe des intervalles de temps calculés est spécifiés dans chaque cas. Le schéma *b* représente l'enchaînement d'événements intervenant dans le scénario de découverte d'origine malveillante. Les schémas *c* et *d* représentent les enchaînements d'événements tels qu'ils peuvent avoir lieu dans le scénario de découverte d'origine non malveillante. En effet, l'événement d'apparition du kit d'exploitation peut avoir lieu avant la publication du correctif (schéma *c*) ou après (schéma *d*).



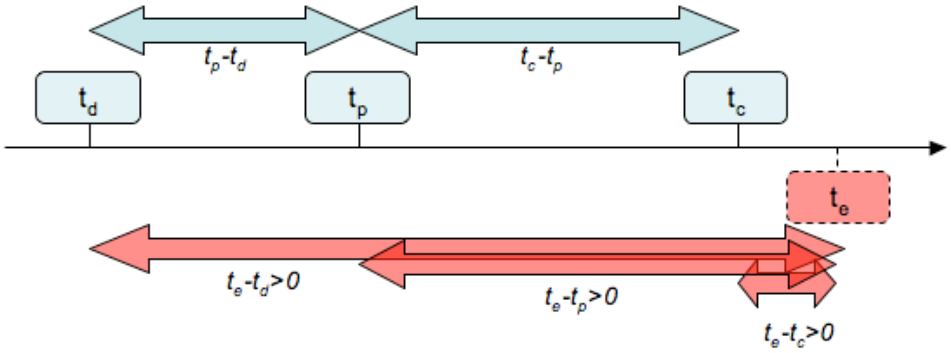
(a)



(b)



(c)



(d)

Figure 4.1 : Scénarios considérant le cycle de vie et l'apparition du kit d'exploitation

Pour commencer, nous étudions ici le phénomène de publication de la vulnérabilité dans un contexte de découverte d'origine non malveillante. Pour cela, nous tenons compte des vulnérabilités pour lesquelles nous connaissons la date de découverte et la date de publication : dans ce cas, nous considérons la durée t_p-t_d, positive ou nulle. Dans les données que nous avons récupérées depuis la base de données de vulnérabilités OSVDB, il y a 3926 vulnérabilités dont les dates de découverte et de publication de la vulnérabilité sont disponibles. Dans cet ensemble de vulnérabilités, 708 vulnérabilités ont été découvertes et publiées le même jour. 3218 vulnérabilités ont été publiées un jour ou plus après leur découverte. La formule de Sturges indique 12 classes pour le traitement des données. L'histogramme, illustré dans la figure 4.2, représente le nombre de vulnérabilités ayant le même délai t_p-t_d entre la découverte et la publication de la vulnérabilité.

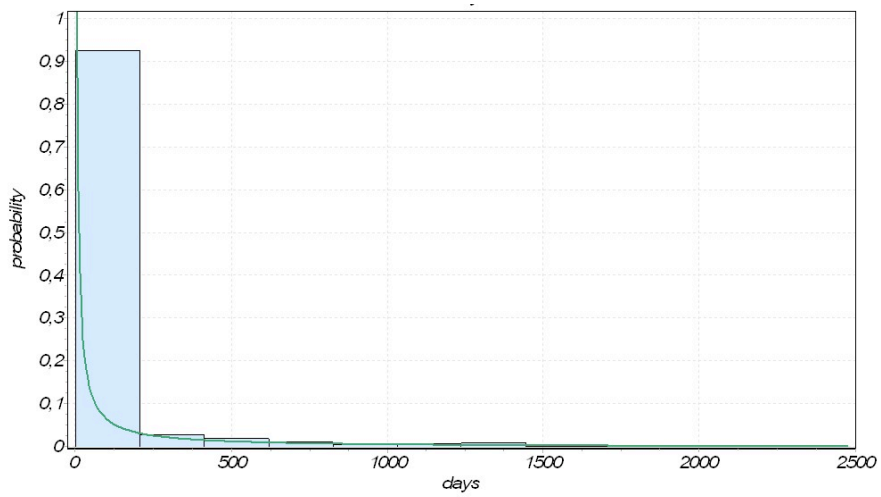


Figure 4.2 : Durée entre événements de découverte et de publication de la vulnérabilité

La distribution Bêta est celle qui concorde le mieux avec les données. De plus, elle est validée par le test de Kolmogorov-Smirnov. Les paramètres sont récapitulés dans le tableau 4.4. Les paramètres α et β sont des paramètres de forme relatifs à la distribution Bêta. Nous rappelons la fonction de densité ci-dessous :

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

Un avantage certain de cette distribution de probabilité et qu'elle peut être facilement utilisée dans notre contexte car elle est prise en compte dans l'outil Möbius.

Maintenant, nous considérons le second scénario dans lequel nous nous basons sur une découverte de la vulnérabilité d'origine malveillante. Pour analyser cet événement, nous avons besoin de prendre en considération les dates de publication t_p et de parution du kit d'exploitation t_e, telles que la différence t_p-t_e ait une valeur positive. Notre ensemble de vulnérabilités se compose de 222 vulnérabilités. L'intervalle de temps entre ces deux

événements peut aller de 1 à 2151 jours. Nous traitons les données sous la forme d'un histogramme composé de 8 classes – nombre indiqué par la formule de Sturges. Ici encore, nous trouvons que la distribution de probabilité Bêta correspond aux données que nous possédons, mais avec des paramètres différents que ceux déterminés auparavant. L'histogramme des données et la fonction Bêta associée sont présentés dans la figure 4.3.

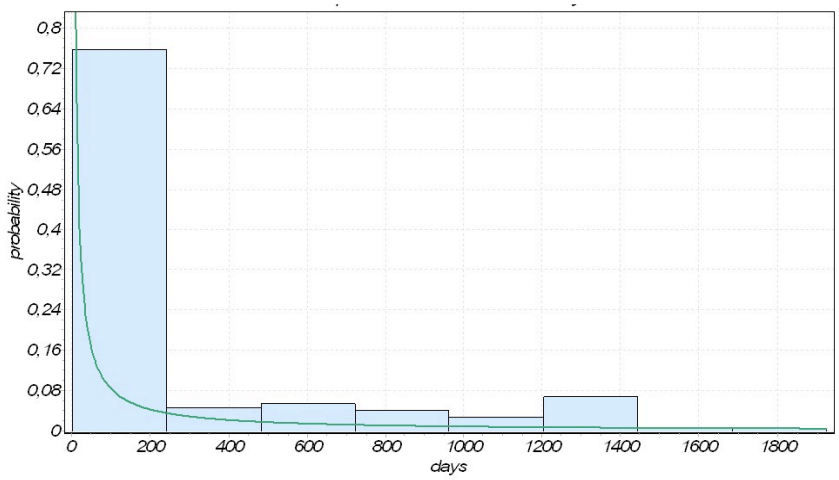


Figure 4.3 : Durée entre événements d'apparition du kit d'exploitation et de publication de la vulnérabilité

1.1.2.4.2 Etude du phénomène de publication du correctif de la vulnérabilité

L'événement de publication du correctif de la vulnérabilité est étudié selon le même processus d'analyse que nous avons utilisé pour caractériser le processus de publication de la vulnérabilité. Notre ensemble de vulnérabilités comporte cette fois-ci 871 vulnérabilités. Pour 712 d'entre elles, le correctif a été publié le même jour que la vulnérabilité. Les intervalles de temps entre publication de la vulnérabilité et publication du correctif vont de 0 à 759 jours. Nous traitons les données sous la forme d'un histogramme comportant 10 classes de données – nombre déterminé par la formule de Sturges. Ici encore, la distribution Bêta convient et est validée par le test de Kolmogorov-Smirnov. Les résultats sont montrés sur la figure 4.4.

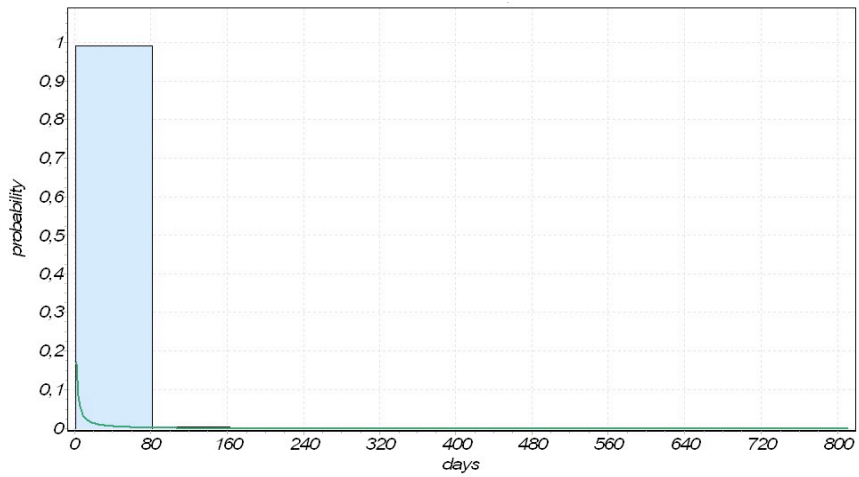


Figure 4.4 : Durée entre événements de publication de la vulnérabilité et publication du correctif

1.1.2.4.3 Etude du phénomène de création du kit d'exploitation

Nous avons expliqué dans les paragraphes précédents que la création du kit d'exploitation n'était pas un événement propre au cycle de vie de la vulnérabilité car il peut survenir à n'importe quel moment, relativement aux événements du cycle de vie de la vulnérabilité. Mais à cause des interdépendances existantes que nous avons identifiées, il nous apparaît tout de même important d'étudier les intervalles de temps entre l'apparition du kit d'exploitation et certains événements du cycle de vie de la vulnérabilité.

Nous commençons par considérer l'ensemble des données tel que la date d'apparition du kit d'exploitation et la date de découverte de la vulnérabilité sont connues. Nous comptons 2108 vulnérabilités correspondantes. Dans cet ensemble de vulnérabilités, 389 vulnérabilités possèdent une date de découverte identique à la date de parution du kit d'exploitation. Dans ce cas de figure, on peut envisager avec certitude que la découverte de la vulnérabilité est d'origine malveillante. Ici encore, la distribution de probabilité Bêta est un modèle acceptable pour les données, et est validée par le test de Kolmogorov-Smirnov.

Maintenant, nous nous concentrons sur la comparaison entre la publication de la vulnérabilité et l'apparition du kit d'exploitation. Cette partie de notre étude est basée sur un ensemble de 17857 vulnérabilités. Dans ce jeu de données, il est à noter que :

- Pour 222 vulnérabilités, le kit d'exploitation est apparu avant la publication de la vulnérabilité – nous avons utilisé ces vulnérabilités pour caractériser le phénomène de publication de la vulnérabilité dans l'hypothèse du scénario avec découverte d'origine malveillante ;
- Nous avons 17077 vulnérabilités pour lesquelles la publication et le kit d'exploitation apparaissent le même jour ;
- Pour 558 vulnérabilités, le kit d'exploitation est apparu après la publication de la vulnérabilité. Pour ces vulnérabilités, nous pouvons considérer que nous sommes dans l'hypothèse d'un scénario de découverte non malveillante.

Il est important de noter la grande quantité de vulnérabilités ayant été publiées et exploitées le même jour. Cela met en évidence la grande importance de l'événement de

publication de la vulnérabilité. Nous nous plaçons dans le cadre du scénario de découverte d'origine non malveillante, c'est-à-dire les deux dernières catégories de données que nous venons d'énoncer. Nous formons 15 classes de données pour procéder à l'analyse, comme indiqué par la formule de Sturges. Cet ensemble de données peut être lui aussi modélisé par une distribution Bêta. Ici encore, cette distribution satisfait le test de Kolmogorov-Smirnov.

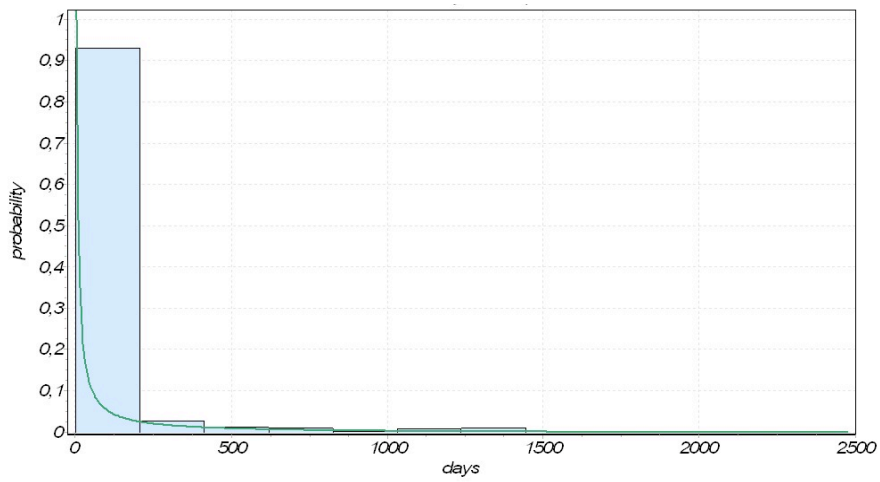


Figure 4.5 : Intervalles de durée entre découverte de la vulnérabilité et l'apparition du kit d'exploitation

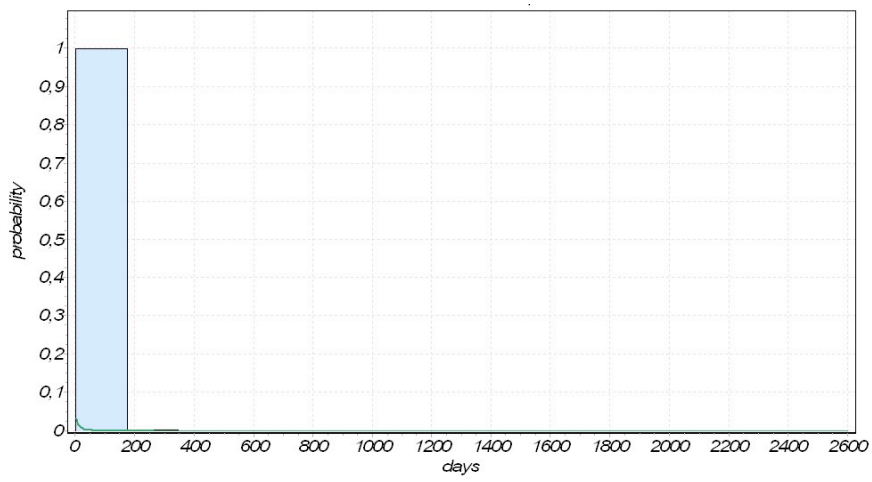


Figure 4.6 : Intervalles de durée entre la publication de la vulnérabilité et l'apparition du kit d'exploitation

Nous rappelons que l'ensemble des paramètres des distributions caractérisant les événements étudiés dans ce paragraphe sont résumés dans le tableau 4.4.

La pente de la distribution de probabilité Bêta augmentant avec le paramètre β, nous pouvons conclure que l'apparition du kit d'exploitation est un événement qui intervient rapidement, que ce soit après la publication de la vulnérabilité dans le cadre d'un scénario de découverte non malveillante ou après le phénomène de découverte dans le scénario de découverte malveillante. Ces données mettent donc en évidence la réactivité très importante de la population des attaquants.

Confrontons ces conclusions à ce que nous avons observé dans l'analyse préliminaire. Nous avons vu que l'ensemble de vulnérabilités possédant une date d'apparition du kit d'exploitation connue représente 34.95% des vulnérabilités ayant une date de publication connue. Les causes de l'absence de date d'apparition de kit d'exploitation pour les autres vulnérabilités peuvent être les suivantes :

- La non validité de l'ensemble des données, comme nous l'avons déjà envisagé pour les données relatives à la publication du correctif ;
- La non exploitation de la vulnérabilité : il n'y a pas de kit d'exploitation existant ;
- Un kit d'exploitation existe mais la date est inconnue ;

La première cause est naturellement envisageable. Néanmoins, il est très difficile d'évaluer l'impact du manque de validité du jeu de données. Nous choisissons donc de partir du principe que l'ensemble des données que nous possédons est un reflet fiable de la réalité. L'explication nous apparaissant la plus cohérente est celle de la non exploitation de la vulnérabilité. Cette explication nous apparaît d'autant plus probable que nous avons observé dans notre analyse que la réactivité des attaquants est très importante : les attaquants produisent un kit d'exploitation très rapidement dans le cas d'une découverte non malveillante. Si une vulnérabilité présente un intérêt important pour la population des attaquants, la probabilité que le kit d'exploitation apparaisse dans un délai inférieur à 24 heures est très élevée. A l'extrême inverse, il nous apparaît également normal qu'il existe une proportion de vulnérabilités sans kit d'exploitation. Il faudra donc tenir compte de cette information dans notre modélisation.

Evénement	Distribution de probabilité adaptée	Valeur de α	Valeur de β
t_p-t_d	Bêta	0.03485	1.6282
t_c-t_p	Bêta	0.00352	0.62362
t_e-t_p	Bêta	0.00090	1.8666
t_e-t_d	Bêta	0.02916	1.5813
t_p-t_e	Bêta	0.03947	0.91506

Tableau 4.4 : Résumé des lois de probabilités et de leurs paramètres

1.1.2.5 Etude par catégorie de vulnérabilités

Nous avons vu dans l'étude précédente que bases de données de vulnérabilités classaient les vulnérabilités enregistrées dans des catégories de vulnérabilités. Celles-ci sont différentes selon la base de données considérée, mais se basent néanmoins sur le type de vulnérabilité (de conception ou de configuration) ou d'attaque permis par la présence de la vulnérabilité sur le système. Dans la base de données OSVDB, il existe 9 classes de vulnérabilités :

- *Input Manipulation* (manipulation des données en entrée) ;
- *Misconfiguration* (configuration du système non voulue rendue possible) ;
- *Denial of Service* (vulnérabilité permettant un déni de service) ;

- *Information Disclosure* (vulnérabilité permettant un viol de la confidentialité des données) ;
- *Race Condition* (vulnérabilité dans des conditions d'accès de processus concurrentiels) ;
- *Cryptographic* (vulnérabilité dans un algorithme de cryptographie) ;
- *Authentication Management* (vulnérabilité permettant de détourner un mécanisme d'authentification) ;
- *Other* : vulnérabilité non correspondant pas à une des catégories citées ;
- *unknown* : catégories non spécifiée.

Nous avons mené le même type d'analyse que celles menées pour la globalité des vulnérabilités considérées pour la base de données. En effet, il est important d'étudier si il n'existerait pas de grosses disparités entre deux types d'attaque : une classe ne comportant que peu de vulnérabilités avec un comportement radicalement différent n'aurait pas influé significativement sur le comportement global étudié dans la partie précédente. Ainsi, il est important de vérifier que le comportement caractérisé précédemment est bien valable pour tous les types de vulnérabilités, ou bien au contraire, de mettre en évidence les différences de comportement.

	Tp-Td	Tc-Tp	Te-Tc	Te-Tp	Te-Td
Input Manipulation	2657	805	255	13682	1444
Misconfiguration	57	9	0	303	142
Denial of Service	403	113	8	1221	197
Information Disclosure	795	93	24	3136	481
Race Condition	53	4	1	152	22
Cryptographic	46	10	3	169	29
Authentication Management	134	25	10	340	68
Other	50	6	0	71	24
Unknown	12	16	0	7	2
PAS DE CLASSE	459	113	6	1310	161
TOTAL	3960	1151	290	17857	2131

Tableau 4.5 : Données par catégories

Chaque catégorie n'a malheureusement pas été analysable, et cela en raison d'un trop petit nombre de données, comme l'indique les données du tableau 4.5. Certaines catégories ne possèdent pas de vulnérabilités : les analyses ont donc été impossibles.

Les analyses ont révélé que, pour chacune des catégories de vulnérabilités analysables, les données correspondaient à une distribution de probabilité suivant une loi Bêta.

Ces analyses n'ont donc pas mis en évidence de comportement singulier vis-à-vis du comportement global que nous avions caractérisé. Ce résultat peut amener à réfléchir sur la pertinence de la classification des vulnérabilités que nous avons utilisée : en effet, les critères utilisés pour cette classification ne sont peut-être pas ceux qui permettent de mettre en valeur des comportements singuliers. De plus, certaines catégories de vulnérabilités ne contenant pas ou peu d'éléments, il est difficile de conclure.

1.2 Conclusion de l'étude des vulnérabilités

Cette étude de vulnérabilités nous permet de quantifier plusieurs événements de notre étude. Dans un premier temps, nous nous sommes penchés sur le problème de l'acquisition et du traitement des données. Nous avons pu obtenir et traiter des données grâce à la base de données OSVDB.

Nous avons procédé à une analyse préliminaire de ces données qui a mis en lumière la faible proportion des vulnérabilités possédant un correctif. Bien que nous puissions mettre en doute la validité de notre jeu de données, plusieurs études nous incitent à penser que le phénomène est bien réel. Il faudra donc en tenir compte dans notre modélisation. Il en va de même pour l'apparition du kit d'exploitation

Enfin, nous avons procédé à l'analyse des données de la base de données. A partir de ces données disponibles, nous avons établi que les différents événements pouvaient tous être modélisés grâce à des distributions de probabilité suivant une loi Bêta. Nous avons déterminé les paramètres correspondants et validé ces études par le test de Kolmogorov-Smirnov.

Ces résultats nous sont utiles pour paramétrer les événements de publication de la vulnérabilité, l'événement de publication de la vulnérabilité et l'événement d'apparition du kit d'exploitation. Dans la seconde partie de ce chapitre, nous présentons les simulations effectuées suite à la caractérisation des événements effectuée dans cette partie.

2 Expérimentations

La caractérisation des événements que nous venons de présenter permet d'obtenir un modèle qui soit plus réaliste car cette étude nous a permis de quantifier précisément la probabilité d'occurrence des trois événements de publication de la vulnérabilité, de publication du correctif et d'apparition du kit d'exploitation. Ainsi, l'étape suivante de notre approche est de valider notre modèle en tenant compte des nouveaux paramètres obtenus dans l'étude décrite dans la partie précédente. Pour cela nous effectuons deux séries de simulations correspondant aux deux scénarios identifiés à partir de l'événement de découverte de la vulnérabilité.

2.1 Préparation du paramétrage

Dans cette partie, nous présentons le paramétrage des deux modèles. Le premier modèle est un modèle correspondant à un scénario de découverte non malveillante. Le second modèle est un modèle correspondant à un scénario de découverte malveillante. Nous appliquons ainsi le paramétrage à l'aide de l'étude de la base de données de vulnérabilités que nous avons détaillée.

2.1.1 Paramétrage du cycle de vie et de l'apparition du kit d'exploitation

Pour paramétrer les divers événements du cycle de vie ainsi que de l'événement d'apparition du kit d'exploitation, nous allons pouvoir nous appuyer sur la caractérisation des événements que nous avons menée dans la première partie de ce chapitre.

2.1.1.1 Scénario de découverte non malveillante

Dès lors que la vulnérabilité a été découverte par une personne non malveillante, elle est susceptible d'être publiée. Nous avons vu que la quasi-totalité des vulnérabilités découvertes étaient publiées par la suite. Nous appliquons à l'activité de publication de la vulnérabilité la distribution de probabilité Bêta avec les paramètres déterminés dans la première partie de ce chapitre.

A partir de l'instant où la vulnérabilité est publiée, deux événements sont possibles : la publication du correctif et l'apparition d'un kit d'exploitation. Pour ces deux phénomènes, nous avons vu qu'il était possible que l'événement n'ait pas lieu. Nous modélisons donc ce nouvel aspect dans notre modèle grâce à la possibilité d'insérer plusieurs cas dans une activité. Ainsi, l'activité de publication du correctif possède deux cas :

- le premier cas est celui déjà décrit dans le chapitre 3, c'est-à-dire l'occurrence de l'événement de publication du correctif. Cet événement suit une loi de probabilité Bêta paramétrée avec les valeurs déterminées par l'étude. Ce premier cas a une probabilité d'occurrence très faible proche de 2%.
- le second cas est le cas de non occurrence de l'événement de publication du correctif. Ce cas est beaucoup plus fréquent que nous l'avions pensé au départ de ces travaux. Il nous est ainsi apparu que le correctif était soit publié dans un laps de temps très court ou bien même nul, après la publication de la vulnérabilité, soit pas du tout. Il est très important de tenir compte de cette nouvelle information et de l'incorporer à notre modélisation. Nous modélisons donc la non publication du correctif de la vulnérabilité par la conservation du jeton dans la place indiquant que la vulnérabilité est publiée. En effet, la présence de ce jeton est importante pour certaines pré-conditions relatives à d'autres activités du modèle. Cependant, il est nécessaire de spécifier que l'activité modélisant l'événement de non publication du correctif a été tirée en changeant la valeur d'une variable booléenne interne.

Ce changement de modélisation est illustré dans la figure 4.7. L'activité temporisée *correction* possède deux cas distincts, représentés par les deux sorties circulaires de l'activité. La probabilité d'occurrence de chacun des cas est paramétrée. La porte d'entrée *IGcorrection* teste la variable booléenne interne. Celle-ci a été initialisée à la valeur FAUX car l'activité *correction* n'a jamais été franchie. La porte d'entrée *IGcorrection* vérifie également la présence d'un jeton dans la place *Vp*. L'activité *correction* sélectionne un cas de sortie en respectant les proportions attribuées. Dans l'hypothèse de sortie correspondant à la porte de sortie *OGcorrection*, le jeton contenu dans la place *Vp* est retiré et ajouté dans la place *Vc*. Le comportement est identique à la précédente modélisation. Dans l'hypothèse où la sortie sélectionnée est celle correspondant à la porte de sortie *IGnoCorrection*, le jeton contenu par la place *Vp* n'est pas déplacé, car ce cas signifie que cette vulnérabilité ne sera pas corrigée. La valeur VRAI est affectée à la variable booléenne afin que l'activité *correction* ne soit plus sensibilisée. Cependant, garder le jeton dans la place *Vp* est essentiel pour la validité de notre modèle.

- Le même principe de modélisation s'applique pour l'événement d'apparition du kit d'exploitation. La probabilité de création du kit d'exploitation est en revanche très supérieure avec une valeur avoisinant les 34,5%.

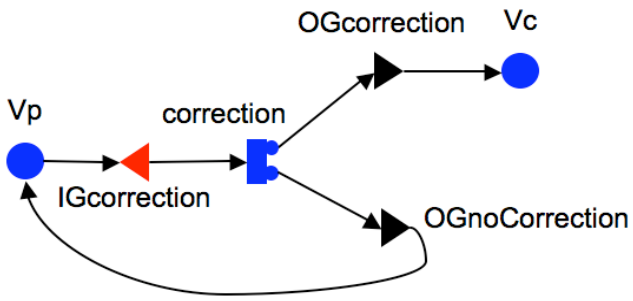


Figure 4.7 : Changement de modélisation appliqué au phénomène de publication du correctif

2.1.1.2 Scénario de découverte malveillante

Dans le cas du scénario de découverte malveillante, la découverte de la vulnérabilité entraîne l'apparition d'un kit d'exploitation.

L'apparition du kit d'exploitation est considérée automatique après la découverte de la vulnérabilité. Donc nous modélisons cette activité avec une distribution de probabilité suivant une loi Bêta et paramétrée avec les valeurs déterminées dans notre analyse.

Cet événement d'apparition du kit d'exploitation est l'élément déclencheur de la publication de la vulnérabilité. L'événement de publication de la vulnérabilité va donc être modélisé avec la distribution de probabilité suivant la loi Bêta et possédant les paramètres évalués lors de notre étude. La présence d'un kit d'exploitation avant même la publication de la vulnérabilité a pour conséquence l'occurrence des événements de publication de la vulnérabilité et la publication d'un correctif. Donc, nous modélisons ces événements avec les distributions de probabilité telles que nous les avons définies et paramétrées dans notre analyse présentée au début ce chapitre.

2.1.2 Paramétrage des actions sur le système

Nous avons par la suite paramétré les actions sur le système. Notre objectif à long terme est de pouvoir caractériser ces activités tel que nous l'avons fait pour les activités de publication, publication du correctif et apparition du kit d'exploitation. C'est une des perspectives de notre travail. Nous avons vu cependant que ces analyses représentent une masse de travail très importante et que nous ne nous sommes pas focalisés sur cet aspect dans le cadre de cette thèse Néanmoins, une perspective est de poursuivre ces analyses pour la caractérisation de ces événements. A l'heure actuelle, nous nous basons donc sur les études déjà publiées pour paramétrer ces activités.

2.1.2.1 Phénomène d'attaque

Le phénomène d'exploitation de la vulnérabilité par les attaquants dépend d'une part du scénario de découverte adopté. Le phénomène d'attaque ne peut avoir lieu durant la phase où la vulnérabilité est découverte que dans le cas d'un scénario de découverte malveillante. Les deux autres phases du cycle de vie de la vulnérabilité – phase de vulnérabilité publiée et phase de vulnérabilité corrigée - sont communes aux deux scénarios. Nous avons vu dans le chapitre 3 que nous avions paramétré les activités correspondant aux phénomènes d'attaque dans une phase du cycle de vie par des distributions de probabilité suivant des lois exponentielles. Nous avions choisi de faire

varier le taux moyen de ces lois en fonction de la phase du cycle de vie que nous considérons.

2.1.2.2 Phénomène d'application du correctif et restauration du système

Nous avons vu que le phénomène d'application du correctif pouvait avoir lieu selon trois états différents pour le système, selon que celui-ci est vulnérable, exposé ou compromis. Nous paramétrons ces activités par des distributions de probabilité suivant une loi normale. Le temps moyen d'application du correctif sera d'autant plus court que le système est en danger.

2.1.3 Résultats

Cette partie présente les résultats des deux simulations que nous venons de présenter. Nous détaillons dans un premier temps les résultats obtenus pour le scénario de découverte non malveillante. Puis, dans une deuxième partie, nous détaillons ceux obtenus pour la simulation du scénario de découverte malveillante.

Pour chacune des courbes, nous mesurons la probabilité de présence du jeton dans la place considérée. Comme nous l'avons vu au chapitre 3, cela se fait facilement dans l'outil de simulation Möbius. Une fonction définie dans le modèle *reward* contrôle la présence du jeton dans la place considérée et retourne 1 dans ce cas. Ces résultats nous permettent d'évaluer les valeurs des mesures que nous avons définies.

2.1.3.1 Scénario de découverte non malveillante

2.1.3.1.1 Comportement de l'environnement

Ce paragraphe présente l'évolution des états du cycle de vie de la vulnérabilité ainsi que l'apparition du kit d'exploitation dans le cadre du scénario de découverte non malveillante. Cette partie a pour but de valider le comportement du modèle vis-à-vis de ces événements. L'instant 0 dans le temps représente le phénomène de découverte de la vulnérabilité.

La figure 4.8 présente l'évolution de la présence d'un jeton dans la place *Vd*. Cette courbe décroît très fortement, comportement motivé par le phénomène de publication de la vulnérabilité. Cette courbe, comme la suivante, permet une validation qualitative du comportement de notre modèle[13].

[13] On peut remarquer une légère inflexion dans la courbe. Cela est du à un biais introduit par l'outil de simulation. Il sera malheureusement répercuté sur l'ensemble des courbes que nous présentons. Nous n'en tiendrons pas compte dans notre analyse.

La figure 4.9 montre l'évolution de la présence d'un jeton dans la place *Vp*. On remarque que la probabilité est très grande car ce modèle tient compte du phénomène de non publication du correctif. On remarque donc que le très fort pourcentage de 98% appliqué au phénomène de non exploitation de la vulnérabilité dans ce scénario est vérifié. La figure 4.10, qui représente l'évolution de la présence d'un jeton dans la place *Vc*, confirme ce comportement avec une asymptote à la valeur de 2%. Il sera intéressant de constater les conséquences sur le phénomène de compromission du système par exploitation de la vulnérabilité.

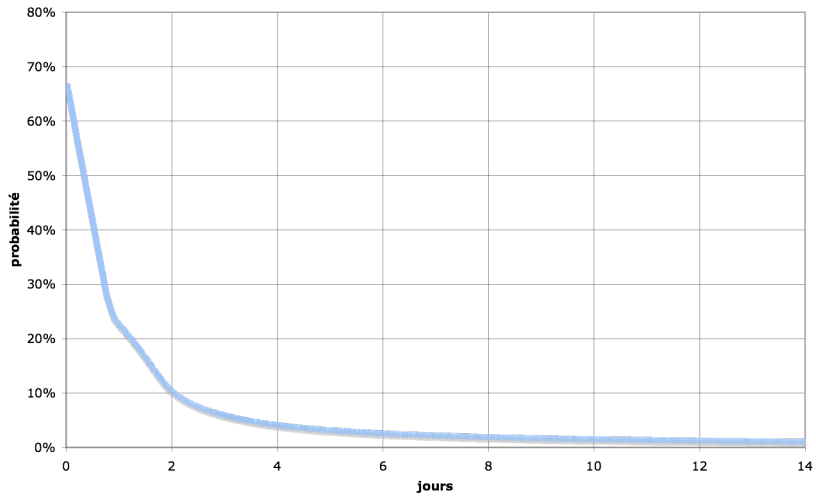


Figure 4.8 : Evolution de la probabilité de présence d'un jeton dans la place *Vd* dans le cadre du scénario de découverte non malveillante

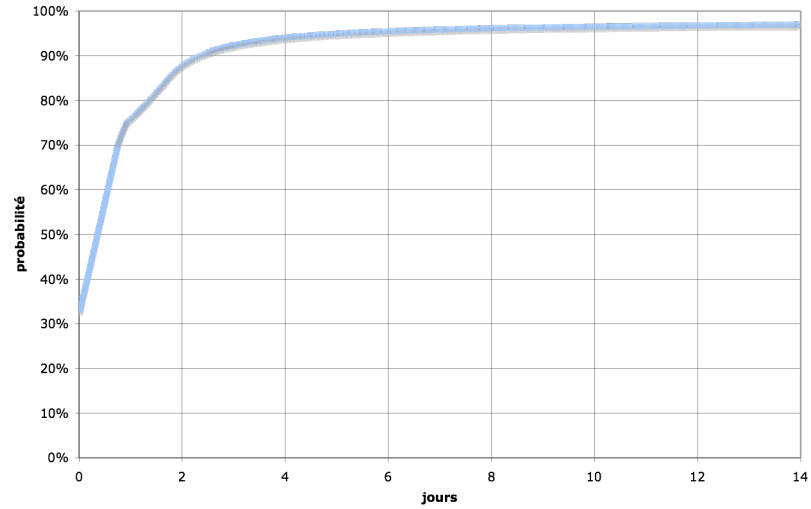


Figure 4.9 : Evolution de la probabilité de présence d'un jeton dans la place *Vp* dans le cadre du scénario de découverte non malveillante

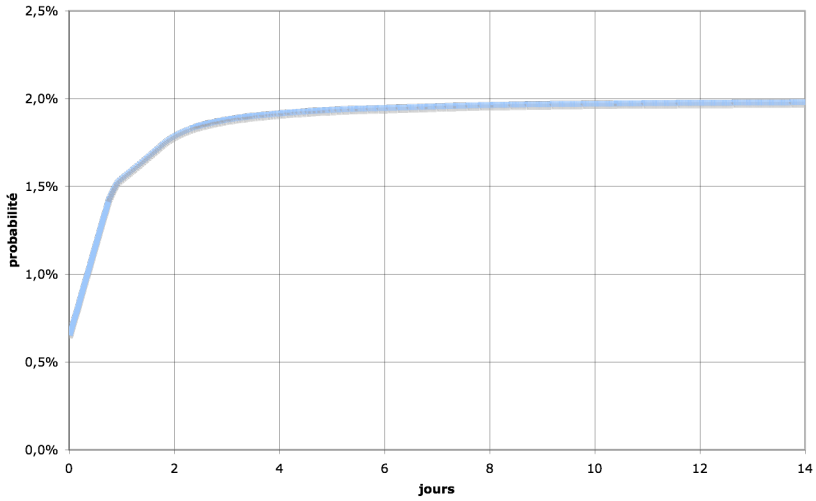


Figure 4.10 : Evolution de la probabilité de présence d'un jeton dans la place *Vc* dans le cadre du scénario de découverte non malveillante

Enfin, la courbe 4.11 montre également un comportement correct de la modélisation de l'environnement, avec l'asymptote de la courbe respectant les 35% paramétrés. On peut remarquer la mise au point rapide du kit d'exploitation qui atteint 30.30% en seulement deux jours. A cet instant, il y a une probabilité de 90.26% que la vulnérabilité ait été publiée. Comme nous l'avons évoqué au début de ce paragraphe, ces courbes ont pour objectif de vérifier le fonctionnement cohérent de notre modèle. Dans la partie suivante, nous détaillons l'évolution des différents états du système qui nous permettent l'évaluation des mesures définies dans les chapitres précédents.

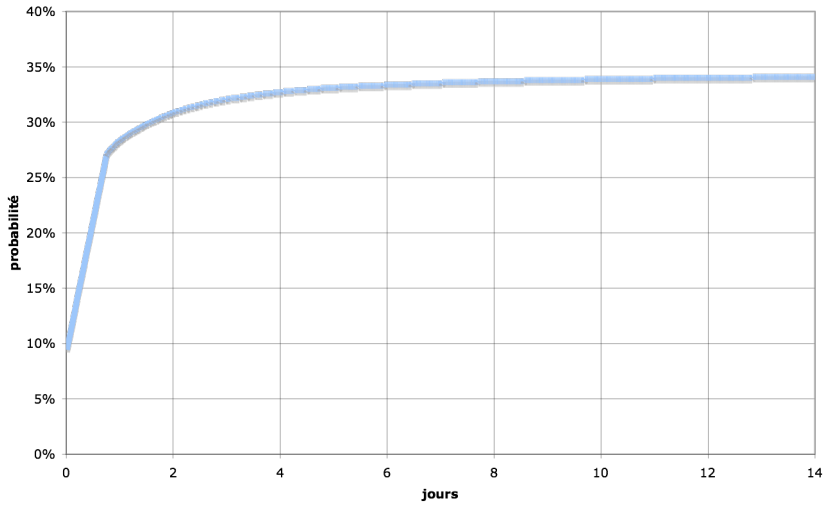


Figure 4.11 : Evolution de la probabilité de présence d'un jeton dans la place *E* dans le cadre du scénario de découverte non malveillante

2.1.3.1.2 Etats du système

Dans cette partie, nous étudions les résultats des simulations en nous focalisant sur l'état du système. Comme nous l'avons vu dans le chapitre précédent, c'est depuis ces résultats que nous allons pouvoir évaluer les mesures que nous avons définies. Nous commençons donc par commenter les courbes obtenues avant d'évaluer les mesures. Dans le cadre de ces expérimentations, nous avons conduit deux expériences simulant deux comportements d'administrateur différents. Les paramètres utilisés sont décrits dans le tableau 4.6 ci-dessous. Les distributions de probabilité utilisées sont les mêmes que celles des expérimentations du chapitre précédent. Notons que nous avons modélisés deux comportements d'administrateur différents – un rigoureux, l'autre non, modélisés par les moyennes des activités *correctionVul*, *correctionExp* et *correctionC*.

Activité	Distribution	Paramètre	Valeur (expérience1/2)
attaqueVp	Exponentielle	taux	10 jours^{-1}
attaqueVc	Exponentielle	taux	5 jours^{-1}
correctionVul	Normale	Moyenne	1 / 30 jours
		Variance	0.5 jours $^{-2}$
correctionExp	Normale	Moyenne	0.5 / 15 jours
		Variance	0.5 jours $^{-2}$
correctionC	Normale	Moyenne	0.01 / 0.5 jours
		Variance	0.5 jours $^{-2}$
reparation	Normale	Moyenne	3 jours
		Variance	0.5 jours $^{-2}$

Tableau 4.6 : Paramètres du modèle pour les activités modélisant les événements agissant sur l'état du système dans le cadre du scénario de découverte non malveillante

La figure 4.12 représente l'évolution de la probabilité de présence d'un jeton dans la place *Vulnerable*. Les deux courbes représentent cette évolution pour les deux comportements de l'administrateur modélisé. La courbe claire représente les résultats pour l'administrateur le plus rigoureux. La courbe foncée représente les résultats pour l'administrateur le moins rigoureux Ces courbes décroissent de manière complémentaire à la croissance des courbes 4.13. Celles-ci représentent l'évolution de la probabilité de présence du jeton dans la place *Expose* qui modélise que le système est exposé à des attaques car il existe un kit d'exploitation disponible. Ces courbes croissent à la mesure de la décroissance de la courbe de la figure 4.12. La décroissance qui suit peut être due à deux événements : l'application du correctif de la vulnérabilité par l'administrateur et une attaque couronnée de succès. Dans le cas des résultats des courbes 4.13, l'événement ayant le plus d'influence est le phénomène d'attaque car celui-ci à un temps moyen d'occurrence bien plus court que celui de l'application du correctif, ce qui explique les deux courbes presque confondues.

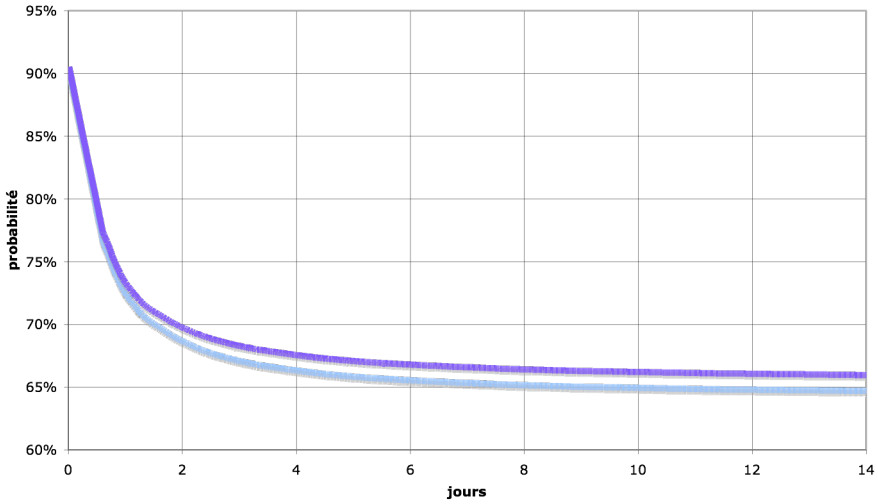


Figure 4.12 : Evolution de la probabilité de présence du jeton dans la place *Vulnerable* à partir de l'instant de découverte de la vulnérabilité

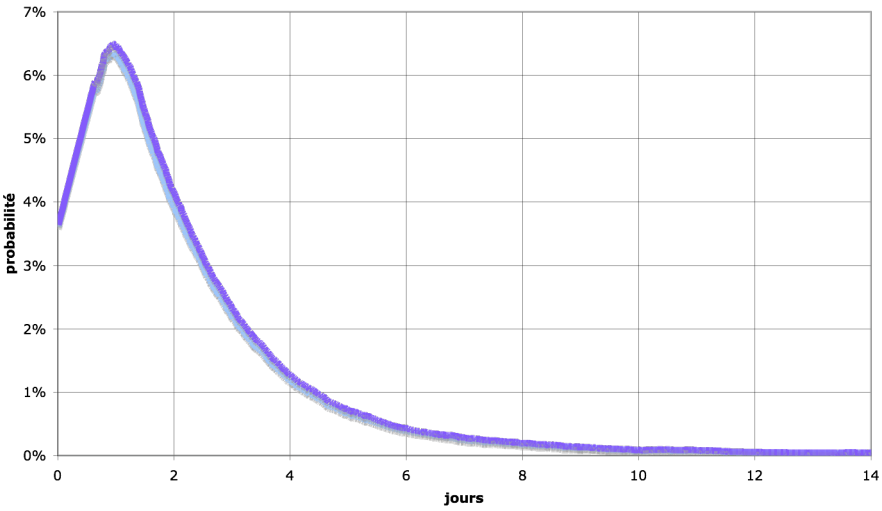


Figure 4.13 : Evolution de la probabilité de présence du jeton dans la place *Expose* à partir de l'instant de découverte de la vulnérabilité

La figure 4.14 montre l'évolution de la probabilité du jeton dans la place *Compromis* modélisant l'état compromis du système. Cette évolution est celle de la mesure PPC(t) que nous avons définie dans le chapitre 2. Nous voyons sur cette courbe que cette probabilité croit rapidement, pour atteindre une asymptote à 34.5%. Cette valeur est la limite de la probabilité d'avoir un kit d'exploitation disponible. Cette asymptote est donc cohérente et donne une mesure maximale du PPC(t) à 34.5% quand t tend vers l'infini. Cependant, on peut remarquer ici, comme sur la courbe précédente, que le comportement

de l'administrateur le plus rigoureux que nous avons modélisé ne suffit pas à compenser la rapidité du phénomène d'attaque. De plus, nous pouvons constater que la probabilité d'être dans un état compromis ne faiblit pas dans le temps. Cela est dû à la très petite probabilité d'avoir un correctif disponible.

La figure 4.15 décrit l'évolution de la probabilité de présence d'un jeton dans la place *Corrige*. Cette place modélise un état transitoire du système entre le moment où celui-ci n'est plus compromis mais pas encore réparé. Le scénario que nous exécutons n'offrant qu'une très petite probabilité d'avoir, à terme, un correctif disponible, la probabilité d'être dans cet état est très faible. Cependant, cette courbe nous permet de mettre en évidence l'influence, même infime dans notre cas, du comportement de l'administrateur. En effet, les deux courbes de la figure 4.15 montrent très nettement les conséquences du comportement de l'administrateur. L'administrateur le plus rigoureux a une probabilité légèrement plus faible d'avoir son système compromis (courbe claire), ce qui diminue la probabilité maximale d'être dans l'état *Corrige* par la suite.

L'addition de cette probabilité avec la probabilité d'être dans l'état compromis permet d'obtenir la mesure PCNR(t) qui représente la probabilité pour le système d'avoir été compromis et d'être encore dans un état où il est susceptible de subir une attaque exploitant cette vulnérabilité. Par exemple, prenons t = 2 jours. Dans le cas de l'administrateur le plus rigoureux, nous avons :

$$PCNR(2\ jours) = 25.70 + 1.19\ .\ 10^{-1} = 25.819\ \%$$

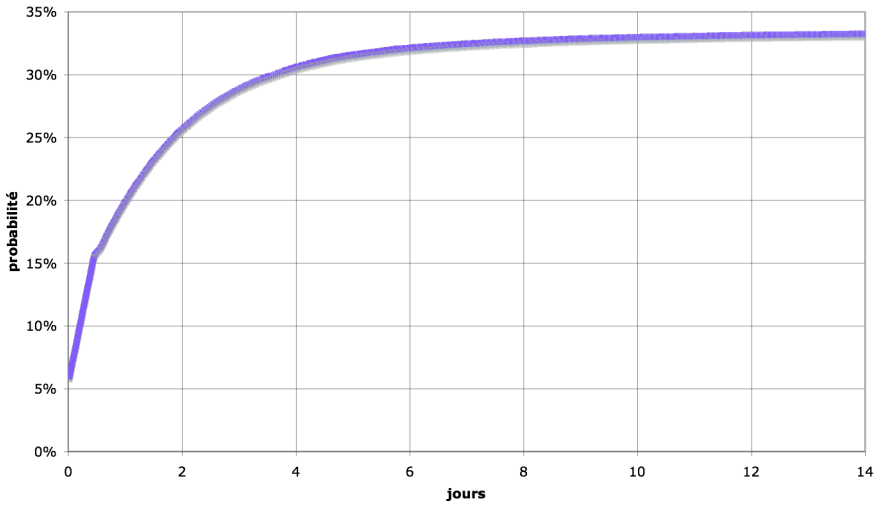


Figure 4.14 : Evolution de la probabilité de présence du jeton dans la place *Compromis* à partir de l'instant de découverte de la vulnérabilité

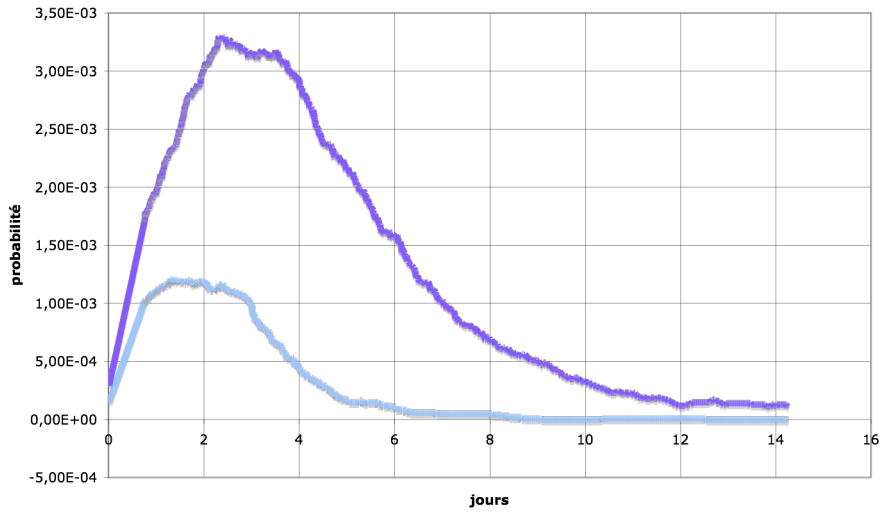


Figure 4.15 : Evolution de la probabilité de présence du jeton dans la place *Corrige* à partir de l'instant de découverte de la vulnérabilité

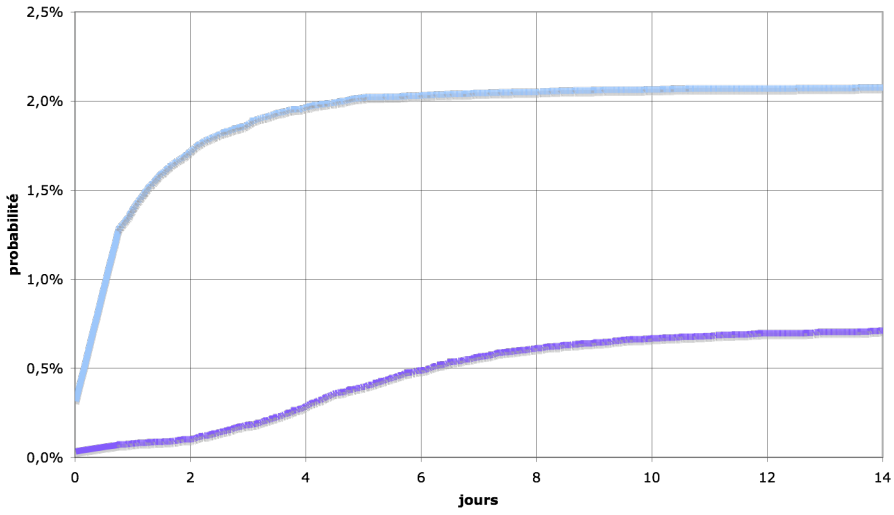


Figure 4.16 : Evolution de la probabilité de présence du jeton dans la place *S* à partir de l'instant de découverte de la vulnérabilité

Ces considérations sont confirmées par les résultats de la courbe 4.16 qui montre l'évolution de la probabilité d'être dans l'état sûr. Dans le cas de l'administrateur le plus rigoureux (courbe claire), la probabilité d'être dans un état sûr est nettement supérieure que dans le cas de l'administrateur le moins rigoureux. Cette dernière courbe représente l'évolution de la mesure PS(t), qui mesure la probabilité que le système soit dans un état sûr vis-à-vis de la vulnérabilité. Ainsi, en considérant l'administrateur le plus rigoureux, on obtient :

PS(2 jours) = 1.65 %

Dans la partie suivante, nous considérons le scénario de découverte malveillante. Comme dans cette partie, nous présenterons d'abord l'évolution de l'état des facteurs de l'environnement avant d'évaluer les valeurs de nos mesures grâce à l'évolution de l'état du système.

2.1.3.2 Scénario de découverte malveillante

2.1.3.2.1 Comportement de l'environnement

Ce paragraphe présente les résultats de l'évolution des états du cycle de vie de la vulnérabilité ainsi que de l'apparition du kit d'exploitation dans le cadre du scénario de découverte malveillante. Comme dans l'étude du scénario précédent, ces premiers résultats ont pour objectif de vérifier la cohérence de notre modèle vis-à-vis des facteurs environnementaux.

La figure 4.17 représente l'évolution de la probabilité de présence d'un jeton dans la place *Vd* de notre modélisation. La présence d'un jeton dans cette place signifie que la vulnérabilité a été découverte mais qu'il n'existe pas encore de kit d'exploitation disponible pour cette vulnérabilité. Cette courbe décroît logiquement de manière très rapide car le phénomène de découverte de la vulnérabilité étant d'origine malveillante, l'apparition du kit d'exploitation intervient très rapidement, provoquant lui-même lui l'événement de publication de la vulnérabilité.

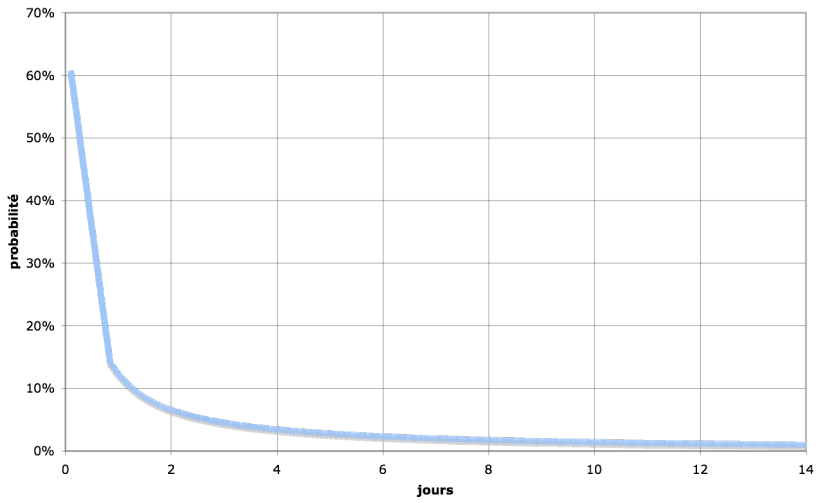


Figure 4.17 : Evolution de la probabilité de présence d'un jeton dans la place *Vd* à partir de l'instant de découverte de la vulnérabilité

L'évolution de la probabilité de présence d'un jeton dans la place *E*, qui traduit l'existence d'un kit d'exploitation est montrée dans la figure 4.18. Une forte probabilité d'existence d'un kit d'exploitation va donc influer sur la probabilité du phénomène de publication de la vulnérabilité. Son évolution est décrite dans la figure 4.19.

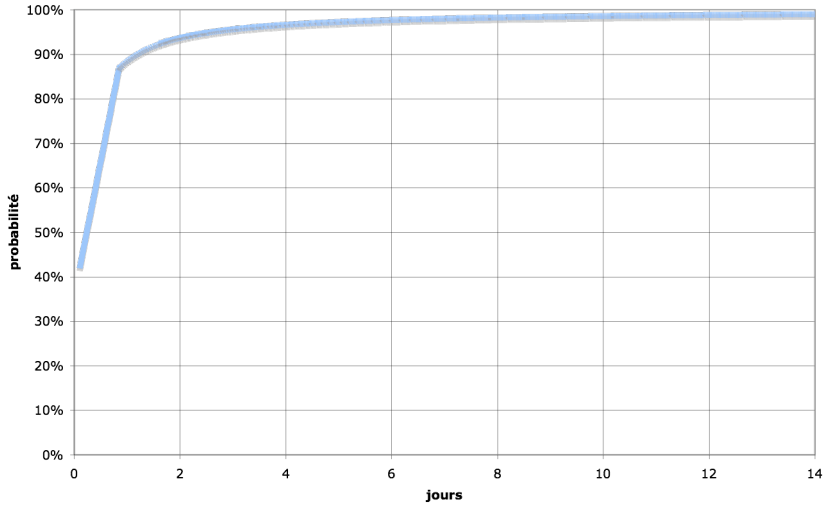


Figure 4.18 : Evolution de la probabilité de présence d'un jeton dans la place *E* à partir de l'instant de découverte de la vulnérabilité

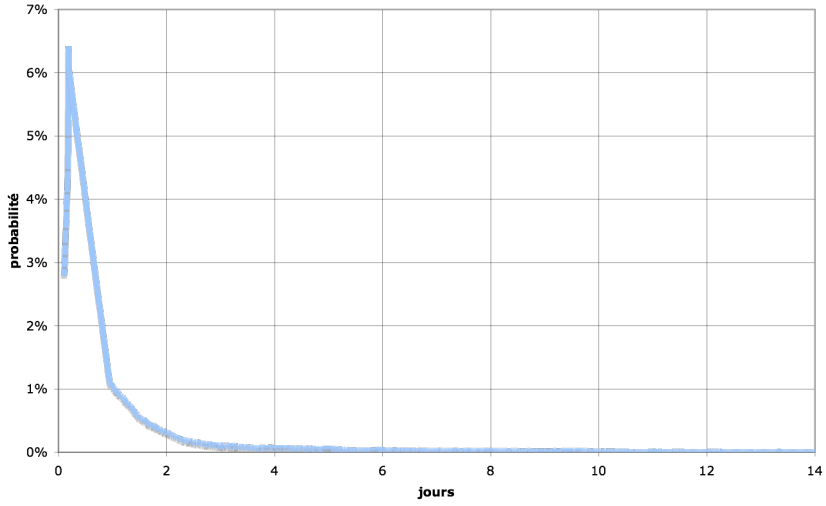


Figure 4.19 : Evolution de la probabilité de présence d'un jeton dans la place *Vp* à partir de l'instant de découverte de la vulnérabilité

La figure 4.19 représente l'évolution de la présence du jeton dans la place *Vp* - modélisant que la vulnérabilité a été publiée mais qu'il n'existe pas encore de correctif disponible. Cette courbe est particulièrement intéressante car l'état "publiée" de la vulnérabilité dépend, dans le contexte du scénario de découverte malveillante, à la fois du comportement des attaquants qui mettent au point un kit d'exploitation et de la publication du correctif de la vulnérabilité. Dans une première phase, la probabilité de publication de la vulnérabilité augmente car la probabilité d'existence d'un kit d'exploitation augmente également. Cependant, l'événement concurrent qui est la publication du correctif fait, par la suite, décroître la probabilité d'être dans l'état où la vulnérabilité est publiée sans être corrigée.

La figure 4.20 représente d'ailleurs l'évolution de la probabilité de présence d'un jeton dans la place *Vc*. Cet état signifie que la vulnérabilité possède un correctif et que l'administrateur peut agir pour protéger son système. La probabilité d'occurrence de cet événement augmente de manière considérable avec la probabilité d'occurrence du phénomène de publication de la vulnérabilité.

Le comportement de notre environnement nous semble cohérent. Dans la partie suivante, nous analysons l'état du système afin de pouvoir déduire les mesures que nous avons présentées dans les chapitres précédents.

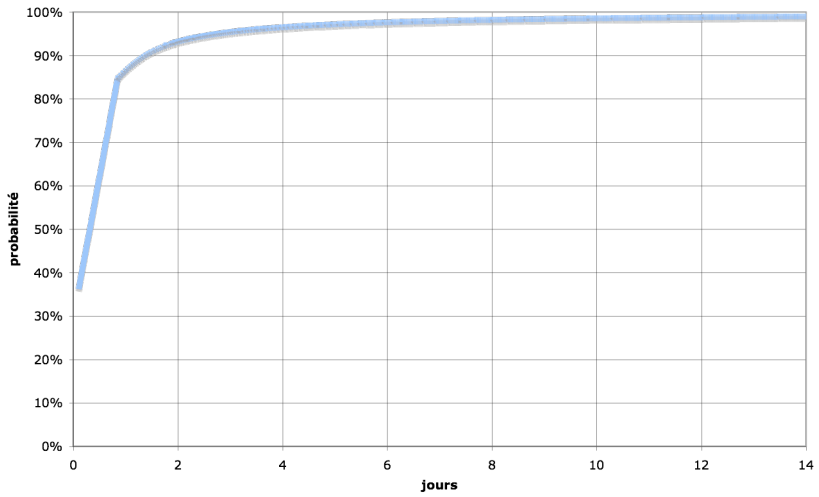


Figure 4.20 : Evolution de la probabilité de présence d'un jeton dans la place *Vc* à partir de l'instant de découverte de la vulnérabilité

2.1.3.2.2 Etats du système

Dans cette partie, nous détaillons l'évolution des probabilités que le système soit dans les différents états : vulnérable, exposé, compromis, non vulnérable et sûr. Dans cette partie, nous nous sommes concentrés sur la mesure PC(t), qui requiert, comme nous l'avons décrit dans le chapitre 3, l'ajout d'une place supplémentaire à notre modèle afin d'être évaluée plus facilement. A partir de ces données, il est possible de calculer les valeurs des mesures que nous voulons. Le tableau 4.7 résume les distributions de probabilité appliquées aux activités agissant sur l'état du système.

Activité	Distribution	Paramètre	Valeur (expérience1/2)
attaqueVd	Exponentielle	Taux	0.01 jours^{-1}
attaqueVp	Exponentielle	Taux	0.05 jours^{-1}
attaqueVc	Exponentielle	Taux	0.01 jours^{-1}
correctionVul	Normale	Moyenne	30 jours
		Variance	0.5 jours^{-2}
correctionExp	Normale	Moyenne	7.0 jours
		Variance	0.5 jours^{-2}
correctionC	Normale	Moyenne	0.6 jours
		Variance	0.5 jours^{-2}
reparation	Normale	Moyenne	1.0 jour
		Variance	0.5 jours^{-2}

Tableau 4.7 : Paramètres du modèle pour les activités modélisant les événements agissant dur l'état du système dans le cadre du scénario de découverte malveillante

Nous commençons par présenter l'évolution de la probabilité de l'état vulnérable, présenté dans la figure 4.21. Il est logique que cette probabilité soit décroissante avec le temps étant donné le phénomène d'apparition du kit d'exploitation qui va amener le système dans l'état exposé, modélisé par la place *Expose*. L'évolution de la probabilité que le système soit dans cet état exposé est décrite dans la figure 4.22. On voit que la probabilité d'être dans l'état exposé augmente au cours d'une première phase qui correspond à l'augmentation de la probabilité d'existence du kit d'exploitation. La décroissance de la courbe s'explique par la suite par l'augmentation de la probabilité d'une attaque couronnée de succès qui fait passer le système dans l'état compromis (figure 4.23).

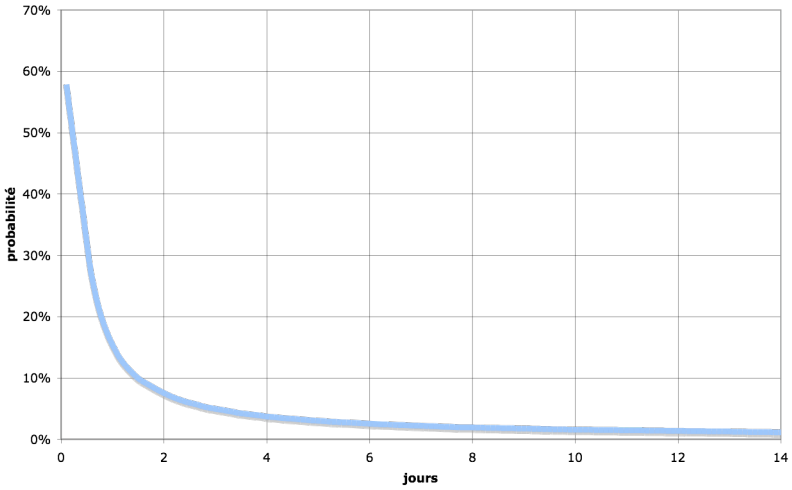


Figure 4.21 : Evolution de la probabilité de présence d'un jeton dans la place *Vulnerable* à partir de l'instant de découverte de la vulnérabilité

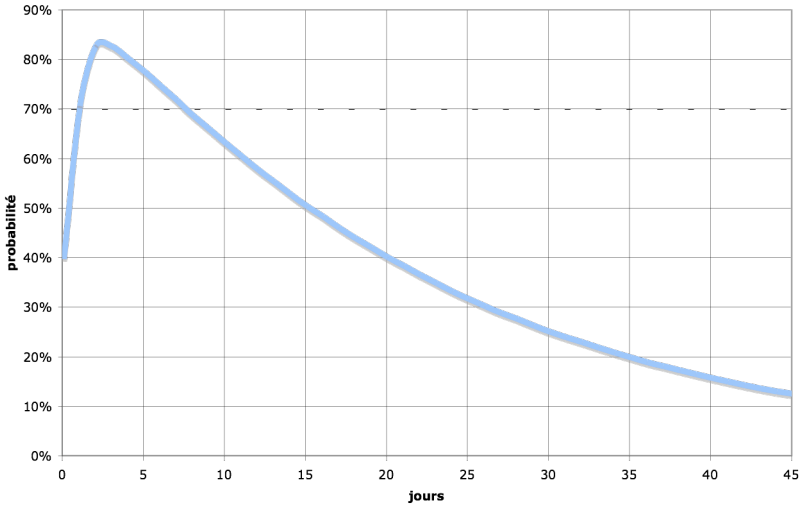


Figure 4.22 : Evolution de la probabilité de présence d'un jeton dans la place *Expose* à partir de l'instant de découverte de la vulnérabilité

La figure 4.23 montre l'évolution de la probabilité de compromission du système par un attaquant exploitant une vulnérabilité. Nous remarquons la tendance de la courbe qui croit très rapidement avec le phénomène de disponibilité du kit d'exploitation. Cependant, on peut remarquer que la probabilité maximale d'avoir le système compromis est faible, ceci à cause du paramétrage d'un administrateur rigoureux et d'un taux d'attaque faible. Le choix de ces paramètres a été fait pour mettre en évidence les influences de chaque facteur environnemental : prendre un taux d'attaque très fort et un administrateur peu rigoureux aurait conduit à une compromission immédiate et n'aurait pas permis de comprendre la tendance de l'évolution de cette probabilité.

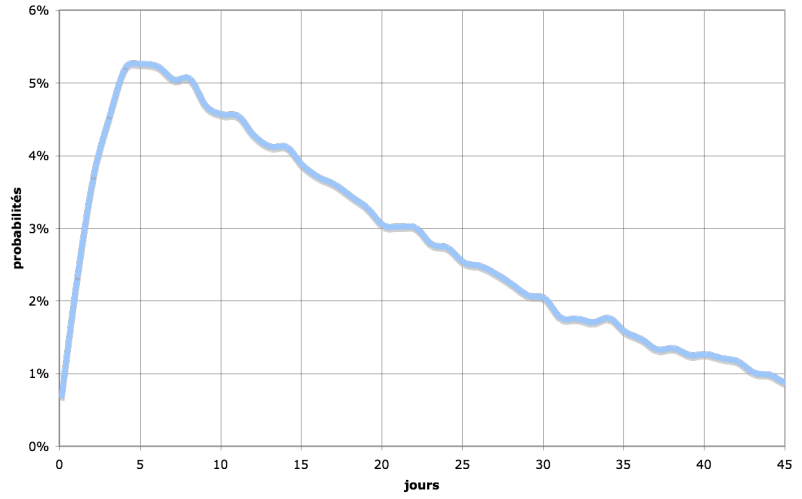


Figure 4.23 : Evolution de la probabilité de présence d'un jeton dans la place *Compromis* à partir de l'instant de découverte de la vulnérabilité – PPC(t)

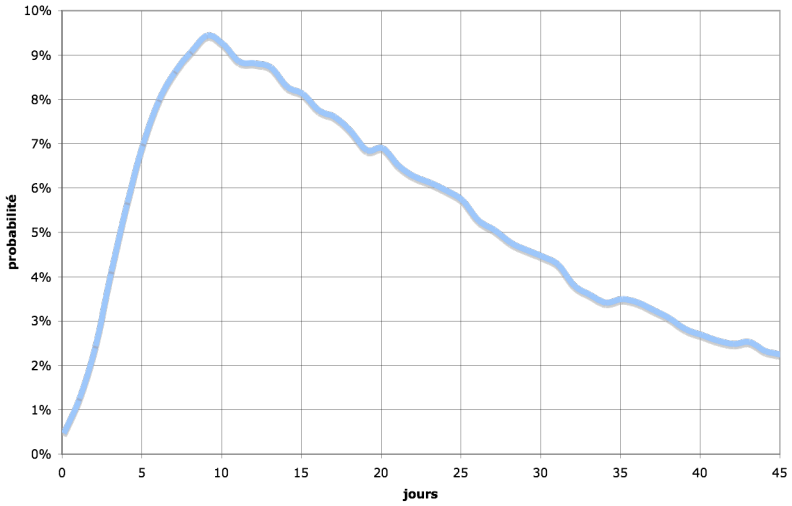


Figure 4.24 : Evolution de la probabilité de présence d'un jeton dans la place *Corrige* à partir de l'instant de découverte de la vulnérabilité

On observe la même tendance pour la courbe décrivant la probabilité que le système soit dans l'état corrigé. Cependant, il est à noter que le pic de la courbe 4.24 est en décalé par rapport à celui de la courbe 4.23. Cela montre donc bien que cet événement fait suite à l'événement de compromission. De plus, nous remarquons que la probabilité maximale d'être dans l'état corrigé est plus grande que celle d'être dans l'état compromis et atteint une valeur de 9.44% au 9^{e} jour contre 5.26% au 6^{e} jour pour l'état compromis, mesuré par PPC(t). Ce constat est intéressant car la seule courbe de l'état compromis aurait pu faire croire à une faible probabilité de compromission du système. Pourtant, la mesure de la probabilité d'être dans l'état corrigé, état du système qui fait suite à l'état compromis, nous montre que la probabilité d'avoir été compromis est plus grande que prévu. La faible probabilité PPC(t) est en réalité due à la compensation du phénomène d'attaque par le phénomène d'application du correctif. La mesure PCNR(t) devient intéressante car elle met en évidence la probabilité bien plus importante que le système soit en danger. En effet, comme cela est montré sur la courbe de la figure 4.25, la mesure PCNR atteint 14.10% entre le 8^{e} et le 9^{e} jour. Ce constat renforce également l'utilité de la mesure PC(t) que nous développons ci dessous.

La figure 4.26 représente l'évolution de la valeur de la mesure PC(t). Cette mesure permet de confirmer la probabilité du danger de compromission du système par l'exploitation de la vulnérabilité. En effet, au bout de 45 jours, la probabilité pour le système d'avoir subi une attaque couronnée de succès s'élève à PC(45 jours) = 87.10%. Cette probabilité est bien supérieure à ce que laisser supposer l'évolution des mesures PPC(t) et PCNR(t), qui représentent un cliché plus "instantané" de l'état du système.

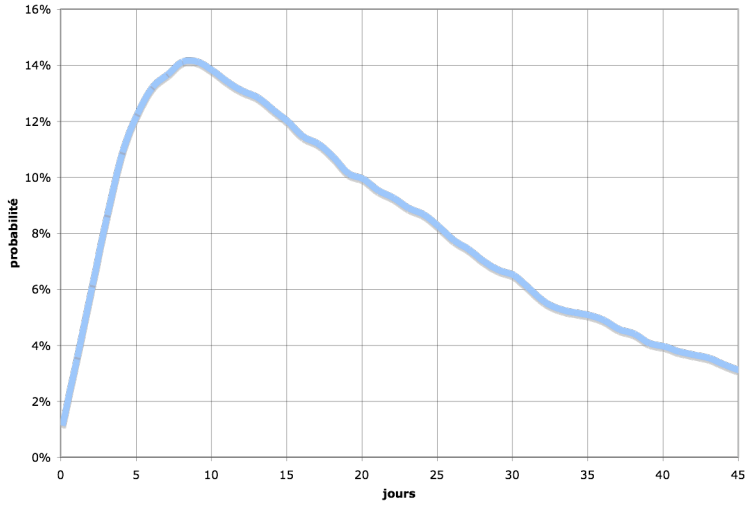


Figure 4.25 : Evolution de la probabilité PCNR(t) à partir de l'instant de découverte de la vulnérabilité

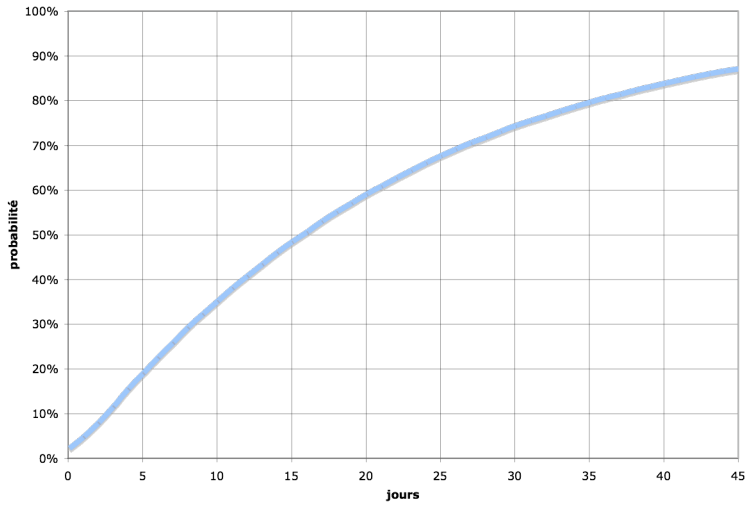


Figure 4.26 : Evolution de la probabilité PC(t) à partir de l'instant de découverte de la vulnérabilité

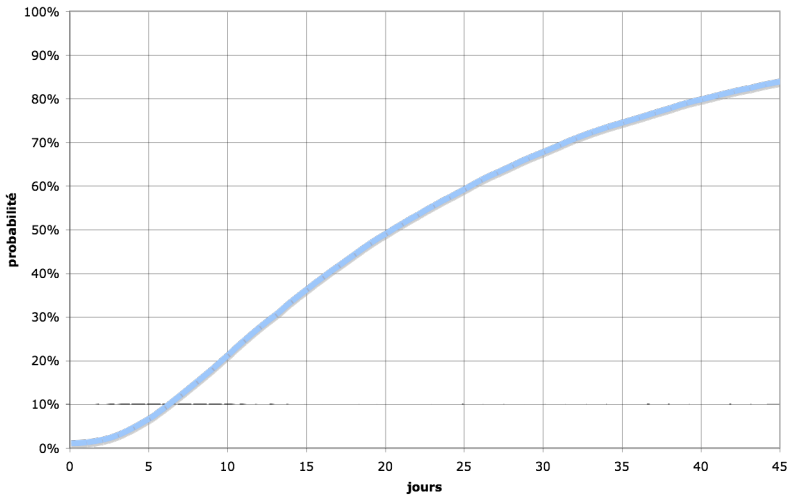


Figure 4.27 : Evolution de la probabilité de présence d'un jeton dans la place *Sur* à partir de l'instant de découverte de la vulnérabilité – PS(t)

Enfin, la figure 4.27 montre l'évolution de la probabilité pour le système d'être dans un état sûr, faisant suite ou non à une compromission. L'ensemble de ces quatre mesures permet d'avoir une vision des risques encourus par le système à plus ou moins long terme et à plus ou moins grande échelle. Les mesures PPC(t) et PCNR(t) sont des mesures permettant un point de vue instantané dans le temps. Leur utilité et de pouvoir connaître la probabilité pour le système d'être dans une configuration précise à un instant donné, vis-à-vis du processus d'exploitation de la vulnérabilité. Les mesures PC(t) et PS(t) permettent un point de vue plus large en examinant un état plus global du système et répondant aux questions : "quelle est la probabilité que mon système subisse une attaque ? "et "quelle est la probabilité pour que mon système soit sûr ?" tout cela en considérant les facteurs environnementaux que nous avons identifiés.

3 Conclusion

Dans ce chapitre, nous avons quantifié les mesures que nous avions spécifiées dans les chapitres précédents. Pour cela, nous avons procédé en deux étapes. La première étape a consisté en la quantification d'événements caractérisant les facteurs environnementaux du cycle de vie de la vulnérabilité (la publication de la vulnérabilité et la publication du correctif) et du comportement de la population des attaquants (l'apparition du kit d'exploitation). Nous avons effectué cette quantification en respectant les deux scénarios d'origine de découverte de la vulnérabilité. Ce travail a été possible grâce à une analyse de la base de données OSVDB.

Dans une seconde phase, nous avons procédé à plusieurs expérimentations. Nos attentes étaient de pouvoir vérifier la cohérence des exécutions du modèle et produire les mesures que nous avions prévues. Dans une première série d'expériences, nous avons étudié le modèle décrivant le scénario de découverte de la vulnérabilité d'origine non malveillante et avons observé deux comportements d'administrateurs. Dans la seconde série d'expérimentation, nous avons étudié le modèle décrivant le scénario de découverte

de la vulnérabilité d'origine non malveillante et avons mis l'accent sur la mesure PC(t), dont l'évaluation requiert un élément de modélisation supplémentaire (cf. chapitre 3). Les résultats de ces expérimentations nous paraissent cohérents. Une de nos perspectives est une étude de sensibilité plus aboutie.

Dans le chapitre suivant, nous présentons les briques de base pour une extension de notre approche afin de pouvoir considérer le processus d'exploitation de plusieurs vulnérabilités.

Chapitre 5 : Extension du modèle à plusieurs vulnérabilités

Dans les chapitres précédents, nous avons établi un modèle permettant de produire des mesures quantitatives de la sécurité du système vis-à-vis d'une vulnérabilité. Malheureusement, il est probable que le système ne comporte pas qu'une vulnérabilité. Il est donc apparu important d'étudier comment étendre le modèle afin de prendre en compte plusieurs vulnérabilités dans le même temps. Mais ce travail demande également d'étudier les interactions qui peuvent exister entre plusieurs processus d'exploitation de vulnérabilités. Ce chapitre présente cette extension et les premières simulations validant cette approche.

1 Extension pour plusieurs vulnérabilités

Bien que notre approche ait été validée, elle présente le désavantage de ne prendre en considération qu'une seule instace de vulnérabilité à la fois. Il est donc important de réfléchir à la prise en compte de plusieurs vulnérabilités dans le même temps.

Comme nous avons un modèle qui fonctionne dans le cadre d'une vulnérabilité, nous visons à étendre notre modélisation par l'exécution en parallèle de plusieurs modèles caractérisant chacun une vulnérabilité (cf. figure 5.1). Chaque modèle est paramétré avec les données quantitatives propres à une vulnérabilité.

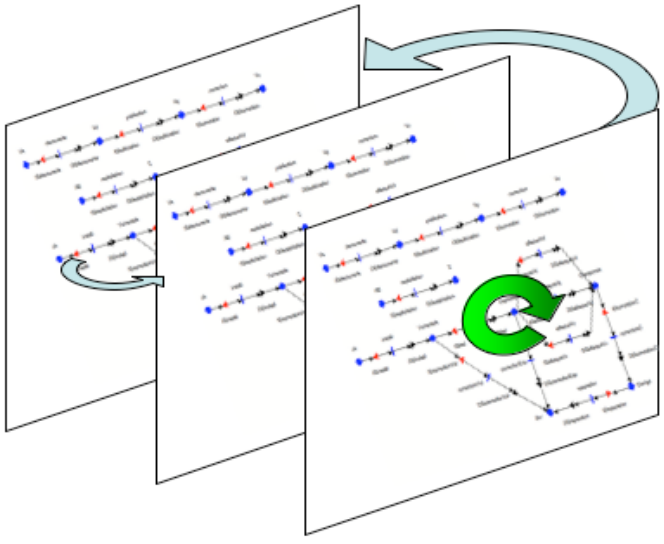


Figure 5.1 : Principe d'extension du modèle à plusieurs vulnérabilités

Cependant, il apparaît très probable qu'il y ait des interactions entre les conséquences de plusieurs processus d'attaque, et donc entre plusieurs modèles s'exécutant en parallèle, comme cela est symbolisé par les flèches claires de la figure 5.1. Nous distinguons deux types de conséquences à la compromission du système par une vulnérabilité :

- l'apparition de nouvelles vulnérabilités exploitables par l'attaquant : c'est-à-dire des vulnérabilités existantes mais pour lesquelles il n'existe pas d'attaquant possédant les privilèges suffisants pour les exploiter ;
- la limitation des actions de certains utilisateurs comme l'administrateur ou encore d'autres attaquants.

1.1 Etude de l'apparition des nouvelles vulnérabilités

L'apparition de nouvelles vulnérabilités peut avoir des causes différentes.

Premièrement, le correctif peut lui-même être un composant vulnérable. Il peut ainsi permettre la correction d'une vulnérabilité tout en introduisant une nouvelle vulnérabilité sur le système. Cette vulnérabilité peut rendre le système exposé ou compromis au même titre qu'une autre. Nous avons pensé à ce cas de figure en incorporant le phénomène d'installation de la vulnérabilité à notre modélisation. Néanmoins, la prise en compte des dépendances entre modèles semble très complexe.

La seconde idée n'est pas relative à l'ajout de vulnérabilité sur le système en lui-même mais se rapproche du principe explicité dans le graphe des privilèges [DAC 94]. C'est-à-dire que l'exploitation d'une vulnérabilité par un attaquant peut lui permettre d'acquérir les privilèges nécessaires pour l'exploitation d'une autre vulnérabilité ; cette vulnérabilité était déjà présente sur le système mais n'était pas exploitable par un attaquant, comme l'illustre l'exemple de la figure 5.2.

Figure 5.2 : Scénario d'exploitation de vulnérabilités

Dans ce scénario, l'attaquant possède tous les privilèges nécessaires pour exploiter la vulnérabilité A présente sur le système. Ce n'est qu'une fois qu'il aura exploité cette vulnérabilité qu'il aura tous les privilèges nécessaires à l'exploitation de la vulnérabilité B. C'est lorsqu'il exploitera cette seconde vulnérabilité que l'attaquant aura atteint le but de son attaque : les dégâts ou les privilèges visés.

Les enchaînements d'exploitation de vulnérabilités forment un processus d'attaque complet visant à atteindre un objectif précis. Cependant, notre intérêt dans cette approche est d'être susceptible de prédire quand il y aura compromission du système, caractérisé par l'exploitation de la première vulnérabilité du processus d'attaque, et non lorsque l'attaquant aura atteint son but, caractérisé par l'exécution totale et avec succès de l'ensemble du

processus d'attaque. Nous nous concentrons sur l'exploitation des vulnérabilités par la population des attaquants, qui est le premier symptôme de ce qui peut être un processus d'attaque plus complexe.

De plus, une telle corrélation entre vulnérabilités est difficile à mettre en évidence dans notre approche. En effet, nous ne considérons pas les vulnérabilités en terme d'impact. Nous nous intéressons à leur profil d'un point de vue temporel. Il est donc impossible, sans intégrer la notion d'impact et de conséquence d'exploitation de la vulnérabilité, de prendre en considération ce type de forte corrélation. Il faudrait connaître précisément les différentes vulnérabilités présentes pour anticiper le processus d'attaque envisagé par un attaquant de la population sur le système et non plus se contenter d'une caractérisation. Cela ne rentrant pas dans le cadre de notre approche, nous décidons de ne pas poursuivre plus loin cette analyse. Dans la partie suivante, nous présentons les interactions entre actions sur le système.

1.2 Les interactions entre actions sur le système

Dans cette section, nous nous intéressons aux dépendances entre sous-modèles de processus de compromission considérant une vulnérabilité. Nous pensons que, à l'image des dépendances entre facteurs environnementaux, les modèles s'exécutant en parallèle doivent avoir une influence mutuelle les uns sur les autres. Il est donc nécessaire d'identifier ces dépendances et de réfléchir au moyen de les modéliser afin d'obtenir une approche complète. Pour cela, nous étudions les deux actions les plus importantes du processus de compromission du système : l'exploitation de la vulnérabilité et l'application du correctif.

1.2.1 L'influence d'une compromission du système

1.2.1.1 Principe et modélisation

Nous nous intéressons donc ici a l'influence que l'état du système peut avoir sur les facteurs environnementaux que nous avons décrits, et plus particulièrement, les actions humaines, malveillantes ou non, possibles avec le système. Nous pouvons donc voir les conséquences de la compromission d'une vulnérabilité comme une interdépendance entre facteurs environnementaux et le système, telle que nous en avons considérée dans notre approche. Mais, à l'inverse de ce que nous avons déjà modélisé, c'est le système qui va influer sur son environnement.

Pour illustrer cela, prenons un scénario d'attaque aux conséquences extrêmes : la défaillance totale du système. Pour cela, considérons deux vulnérabilités sur le système :

- La première, la vulnérabilité A, entraîne, si elle est exploitée avec succès, une défaillance totale du système – le système n'est plus disponible ;
- La seconde, la vulnérabilité B, n'entraîne pas ce type de défaillance mais ne fournit à l'attaquant que des privilèges supplémentaires.

Notons que l'exploitation de ces vulnérabilités est indépendante et que les attaques exploitant A ou B peuvent ainsi être effectuées par des attaquants sans lien entre eux. Deux scénarios sont possibles et sont illustrés dans la figure 5.3.

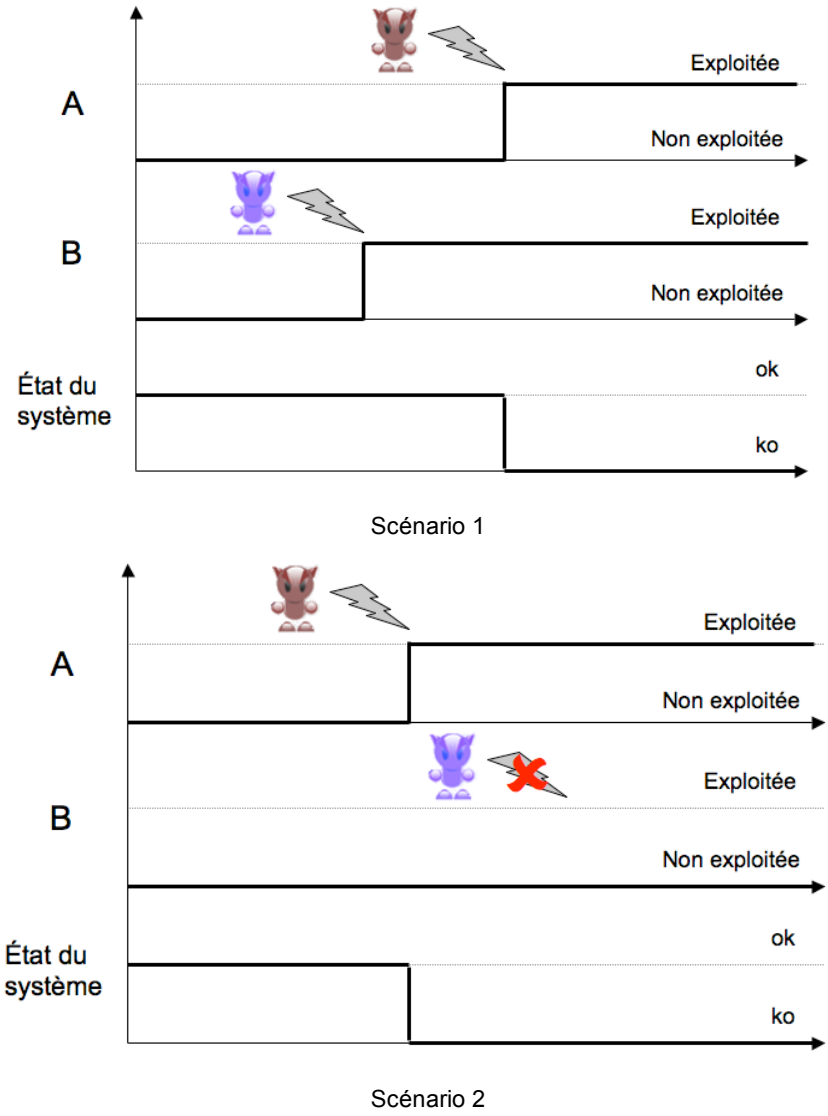


Figure 5.3 : Considération des dépendances entre deux processus d'exploitation de vulnérabilité

Dans le premier scénario, un attaquant exploite la vulnérabilité B. Il obtient les privilèges mais cela ne bloque en rien les actions sur le système. Il est toujours possible d'exploiter la vulnérabilité A. Plus tard, un attaquant exploite cette vulnérabilité A et met le système en état de défaillance. Dans ce scénario, il n'y a pas eu d'interaction entre les deux attaques : l'attaque exploitant la vulnérabilité B n'a pas eu d'influence sur l'exploitation de la vulnérabilité A.

Dans le second scénario, la première vulnérabilité exploitée est la vulnérabilité A. L'attaquant provoque ainsi une défaillance du système. A partir de cet instant, aucune action humaine, qu'elle vienne de l'administrateur ou d'un attaquant, n'est possible. Il est donc impossible pour un attaquant d'exploiter la vulnérabilité B. Il y a donc une forte dépendance entre les deux attaques car le succès de l'exploitation de la vulnérabilité A empêche l'exploitation de la vulnérabilité B.

Ces deux scénarios mettent donc en évidence le besoin de prendre en considération les dépendances créées par l'exploitation de vulnérabilités sur les actions possibles sur le

système. Ces actions sont à la fois celles de la population des attaquants mais aussi celles de l'administrateur.

Dans notre modélisation, on compte plusieurs actions humaines sur le système. L'action des attaquants se résume au phénomène d'exploitation d'une vulnérabilité selon sa phase du cycle de vie. L'administrateur peut, en revanche, 1) appliquer le correctif d'une vulnérabilité selon les états du système, 2) réparer le système après une compromission et 3) installer un nouveau composant vulnérable.

Nous définissons pour commencer trois niveaux de gravité de conséquence de l'exploitation d'une vulnérabilité :

- C1 : il n'y a pas de conséquences de l'exploitation de la vulnérabilité sur les actions possibles ;
- C2 : suite à l'exploitation de la vulnérabilité, la seule action possible est l'application du correctif de la vulnérabilité exploitée ;
- C3 : suite à l'exploitation de la vulnérabilité, plus aucune action n'est possible, le système a subi une défaillance et ne remplit plus sa fonction.

Ces trois niveaux de gravité ne composent pas une liste exhaustive et il serait possible d'en définir de nombreux autres. En fait, toutes les combinaisons logiques d'actions humaines sont des niveaux de gravité de conséquence modélisables. Néanmoins, nous limitons notre modélisation aux trois degrés de gravité présentés.

L'utilisation de ces degrés de gravité a le principe suivant : dès l'instant où une vulnérabilité a été exploitée avec succès, il faut que les actions entravées par son exploitation soient rendues impossibles, dans le modèle de la vulnérabilité exploitée ainsi que dans les autres modèles de vulnérabilité s'exécutant en parallèle. Pour le degré de gravité C1, aucune action n'est rendue impossible, autant dans le modèle de la vulnérabilité exploitée que dans ceux s'exécutant en parallèle. Le degré de gravité C2 interdit toutes les actions, mise à part l'action d'application du correctif dans le modèle de la vulnérabilité exploitée. Enfin, le degré de gravité C3 interdit toutes les actions. Il faut donc une communication de l'information de la compromission du système par l'exploitation d'une vulnérabilité entre les différents modèles qui s'exécutent. Voyons comment cela peut être modélisé.

Nous avons vu au début de ce chapitre que les SAN étaient un formalisme de modélisation très puissant permettant de combiner plusieurs modèles pour former un modèle composé. Cela est rendu possible en particulier grâce à l'opérateur JOIN. De plus, la composition de modèles inclut le partage de places. Ce sont ces deux propriétés qui vont nous permettre de modéliser notre approche à plusieurs vulnérabilités simultanées.

Pour chaque degré de gravité, on définit une place partagée. Ayant défini les trois degrés de gravité C1, C2 et C3, nous ajoutons au modèle deux nouvelles places : *C2* et *C3*. Ces deux places seront partagées dans chacun des modèles s'exécutant en parallèle. Notons que nous ne créons pas de place *C1* : en effet, considérant que ce degré de gravité traduit l'absence de conséquences, il n'apparaît pas utile de définir une place associée. Les deux places *C2* et *C3* apparaîtront donc dans chaque modèle de processus d'exploitation de vulnérabilité. Ceci est illustré par la figure 5.4.

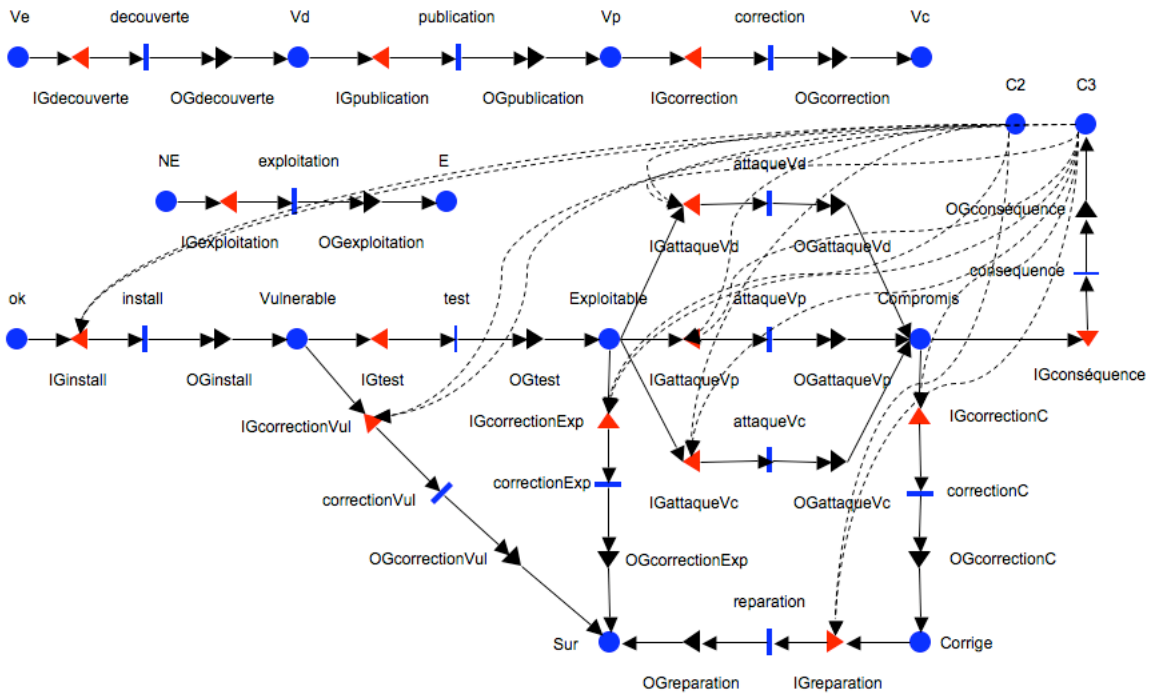


Figure 5.4 : Modèle du processus d'exploitation d'une vulnérabilité avec un degré de gravité de compromission C3 (scénario de découverte d'origine malveillante)

Cette figure représente le modèle du processus de compromission d'une vulnérabilité de degré de gravité C3. Nous avons choisi d'illustrer la modélisation du degré de gravité avec une vulnérabilité découverte par une personne d'origine malveillante ; le principe fonctionne également avec le second scénario. La figure 5.4 montre le modèle tel que nous l'avons simplifié en enlevant certains des arcs symbolisant la vérification de la présence de jeton dans une place. Nous y avons ajouté les deux places partagées *C2* et *C3*, qui modélisent les deux degrés de gravité de compromission du même nom. Dans le modèle, nous avons considéré une vulnérabilité de degré de gravité C3. L'activité *consequence* est une activité instantanée. Elle possède la porte d'entrée *IGconsequence* qui contrôle la présence d'un jeton dans la place *Compromis*. La porte de sortie *OGconsequence* ajoute un jeton dans la place *C3*. Il est à noter que le marquage de la place *Compromis* n'est pas modifié, ce qui empêcherait la bonne exécution du modèle.

La figure 5.4 montre en pointillés les nombreuses vérifications de présence de jeton dans les places *C2* et *C3*. Les pré-conditions contenues dans les portes d'entrée s'en trouvent ainsi modifiées. Ils sont détaillés dans le tableau 5.1.

Les pré-conditions diffèrent de manière générale par l'ajout de la vérification de l'absence de jeton dans les places *C2* et *C3*. Il faut cependant remarquer que la pré-condition contenue dans la porte d'entrée *IGcorrectionC* est spécifique pour chaque degré de gravité qui peut être affecté à la vulnérabilité.

Porte d'entrée	Prédicat	
IGattaqueVd	(marquage(C3)==0) ET (marquage(C2)==0) ET (marquage(E)==1) ET (marquage(Vd) == 1) ET (marquage(Expose) == 1)	
IGattaqueVp	(marquage(C3)==0) ET (marquage(C2)==0) ET (marquage(E)==1) ET (marquage(Vp) == 1) ET (marquage(Expose) == 1)	
IGattaqueVc	(marquage(C3)==0) ET (marquage(C2)==0) ET (marquage(E)==1) ET (marquage(Vc) == 1) ET (marquage(Expose) == 1)	
IGcorrectionVul	(marquage(C3)==0) ET (marquage(C2)==0) ET (marquage(Vc)==1) ET (marquage(Vulnerable) == 1)	
IGcorrectionExp	(marquage(C3)==0) ET (marquage(C2)==0) ET (marquage(Vc)==1) ET (marquage(Expose) == 1)	
IGcorrectionC	Si vulnérabilité de degré C1	(marquage(C3)==0) ET (marquage(C2)==0) ET (marquage(Vc)==1) ET (marquage(Compromis) == 1)
	Si vulnérabilité de degré C2	(marquage(C3)==0) ET (marquage(C2)==1) ET (marquage(Vc)==1) ET (marquage(Compromis) == 1)
	Si vulnérabilité de degré C3	Marquage(C3)==0 (**ce ne sera jamais vrai**)
IGinstall	(marquage(C3)==0) ET (marquage(C2)==0)	
IGreparation	(marquage(C3)==0) ET (marquage(C2)==0)	

Tableau 5.1 : Prédicats des portes d'entrée qui considèrent les places partagées *C2* et *C3*

1.2.1.2 Influence sur les mesures

Ces interactions entre les sous-modèles va avoir une influence sur les mesures que nous avions définies dans les chapitres précédents. Les valeurs de ces mesures seront donc évaluées en considérant l'ensemble des sous-modèles. Certaines mesures sont facilement applicables à ce modèle composé.

La mesure PPC(t) évalue la probabilité pour le système d'être dans un état compromis et d'être encore susceptible d'être attaqué. Nous avons vu que, dans le modèle considérant une vulnérabilité, cela se traduisait par la présence d'un jeton dans la place *Compromis*. Dans ce chapitre, la mesure PPC(t) peut se définir l'union des mesures PPC(t) associées à chaque sous-modèle. Cela peut également être définit comme la somme des probabilités *pC1(t)*, *pC2(t)* et *pC3(t)* de présence d'un jeton respectivement dans les places *C1*, *C2* et *C3*, les trois événements étant indépendants.

$$PPC(t) = pC1(t) + pC2(t) + pC3(t)$$

Dans le contexte d'un modèle composé, la mesure PCNR(t) évalue la probabilité qu'un attaquant ayant exploité une des vulnérabilités avec succès soit susceptible de posséder des privilèges ou d'avoir provoqué des dégâts non encore réparés sur le système. Il s'agit également de l'union des ensembles des états compromis et corrigé de chaque sous-modèles. Nous la définissons donc de la manière suivante :

$$PCNR(t) = \bigcup PCNR_n(t)$$

n : nombre de sous-modèles

Il en va de même pour la mesure PC(t) qui se définit comme suit :

$$PC(t) = \bigcup PC_n(t)$$

n : nombre de sous-modèles

Dans le modèle considérant une vulnérabilité, la mesure PS(t) représente la probabilité pour le système d'être dans un état sûr vis-à-vis de la vulnérabilité. Dans le contexte du modèle composé, sa signification s'étend à la totalité du modèle composé. Cette mesure évalue la probabilité que le système soit sûr vis-à-vis de toutes les vulnérabilités considérées. Elle a donc la définition suivante :

$$PS(t) = \prod PS_i(t)$$

n : nombre de sous-modèles

1.2.2 L'influence de l'application d'un correctif

Le phénomène d'attaque n'est pas la seule action humaine susceptible d'avoir une influence sur l'état du système vis-à-vis des autres vulnérabilités prises en compte dans les modèles s'exécutant en parallèle. En effet, l'action principale de l'administrateur, l'application du correctif de la vulnérabilité, exploitée ou non, peut avoir une influence sur l'application des autres correctifs disponibles.

Pour illustrer ceci, considérons le scénario suivant : le système a été compromis par une vulnérabilité exploitée avec succès. Suite à cette attaque, l'administrateur va tout naturellement appliquer le correctif dans la mesure où celui-ci est disponible. Cependant, ne risque-t-il pas de chercher à mettre à jour son système et ainsi corriger les autres vulnérabilités ayant un correctif disponible ?

Nous distinguons donc deux comportements de l'administrateur : 1) si l'administrateur est conscient des dangers liés à la sécurité, l'application d'un correctif entraînera l'application de tous les correctifs disponibles ; 2) dans le cas contraire, il n'appliquera que le correctif de la vulnérabilité connue. Nous pouvons attribuer différentes probabilités correspondant à la probabilité d'occurrence de chacun de ces deux comportements. Ces probabilités peuvent être néanmoins influencées par le contexte dans lequel l'application du correctif est effectuée, selon que le système est vulnérable, exposé ou compromis.

Dans la modélisation, l'influence de l'application du correctif se traduit par l'ajout d'une activité instantanée parallèlement à chaque activité d'application du correctif *correctionVul*, *correctionExp* et *correctionC*. Ces trois activités instantanées ont pour conditions de sensibilisation 1) la disponibilité du correctif, c'est-à-dire la présence d'un jeton dans la place *Vc* et 2) la présence d'un jeton dans une place partagée indiquant que l'administrateur est en train de mettre à jour l'ensemble de son système. Une seconde place partagée modélise l'état où l'administrateur n'est pas en train de mettre à jour son système. L'activité traduisant l'état de non mise à jour vers la mise à jour doit être instantanée et posséder une porte d'entrée n'autorisant la transition que lors de l'application d'un correctif. L'activité modélisant la transition inverse doit être, par contre, temporisée. Il est possible

d'appliquer une probabilité de transition de l'activité menant vers l'état de mise à jour ou d'ajouter un prédicat en fonction de la gravité de la compromission.

Bien que nous sachions comment modéliser ces aspects, nous avons choisi de ne pas l'implémenter dans notre modèle pour l'instant, dans le but de ne pas complexifier nos expérimentations futures. Néanmoins, ces comportements nous apparaissaient suffisamment importants pour être étudiés.

2 Expérimentations

Dans cette partie, nous présentons les premiers résultats obtenus par la simulation d'un modèle étendu. Le formalisme des SAN que nous avons adopté pour notre modélisation permet de composer les modèles pour chaque vulnérabilité et de les composer pour les exécuter ensemble en tenant compte de leurs dépendances. Le simulateur Möbius avec lequel nous faisons nos expérimentations est capable d'éxécuter des simulations pour un modèle composé d'un nombre important de modèles mais requiert une puissance de calcul tout aussi importante. Cependant, le paramétrage de chaque modèle pour chaque instance de vulnérabilité considéré peut s'avérer compliqué, nous développerons cela dans les perspectives de notre travail.

Dans ce chapitre, nous avons choisi de simuler un modèle tenant compte de deux vulnérabilités. La première est une vulnérabilité de degré de gravité C3 respectant un scénario de découverte non malveillante. La seconde vulnérabilité est une vulnérabilité de degré de gravité C2 respectant un scénario de découverte malveillante. Ces simulations ont pour objectif de valider notre approche.

2.1 Présentation de la simulation

Dans cette partie, nous présentons les paramètres que nous appliquons à notre simulation. Pour plus de lisibilité, nous présentons d'abord les paramètres appliqués à la vulnérabilité de degré de gravité C3, puis ceux appliqués à la vulnérabilité de degré de gravité C2.

2.1.1 Paramétrage de la vulnérabilité de degré de gravité C3

Le degré de gravité C3 associé à la vulnérabilité signifie qu'une compromission du système par l'exploitation de cette vulnérabilité empêche toute action d'application de correctif ou de réparation mais aussi tout phénomène d'attaque.

A cette vulnérabilité, nous avons associé le scénario de découverte non malveillante. Les probabilités des événements de publication de la vulnérabilité, de la publication du correctif et de l'apparition du kit d'exploitation sont caractérisés avec des distributions Bêta paramétrées avec les valeurs que nous avons déterminées dans le chapitre précédent.

Nous avons gardé les mêmes distributions de probabilité pour caractériser les phénomènes d'attaque, d'application du correctif. Le phénomène de réparation du système n'a pas a être paramétré étant donné le degré de gravité associé à la vulnérabilité. Les paramètres appliqués à ces distributions de probabilité sont résumés dans le tableau 5.2.

Activité	Distribution	Paramètre	Valeur
attaqueVp	Exponentielle	Taux	0.05
attaqueVc	Exponentielle	Taux	0.01
correctionVul	Normale	Moyenne	30
		Variance	0.5
correctionExp	Normale	Moyenne	7.0
		Variance	0.5
correctionC	Normale	Moyenne	0.6
		Variance	0.5
reparation	Normale	Moyenne	1.0
		Variance	0.5

Tableau 5.2 : Paramétrage du modèle de la vulnérabilité de degré de gravité C3 respectant le scénario de découverte d'origine non malveillante

2.1.2 Paramétrage de la vulnérabilité de degré de gravité C2

Le degré de gravité C2 associé à la vulnérabilité signifie que, lors de la compromission du système par l'exploitation de la vulnérabilité, la seule l'application possible est l'application du correctif.

A cette vulnérabilité-ci, nous avons associé le scénario de découverte malveillante. Nous avons donc associé les paramètres que nous avons déterminés dans le chapitres précédents aux événements de publication de la vulnérabilité, de publication du correctif et d' apparition du kit d'exploitation.

Comme pour la vulnérabilité que nous avons décrite dans le paragraphe précédent, nous avons appliqué les distributions de probabilité adoptées dans les précédentes simulations pour décrire les phénomènes d'attaque, d'application du correctif et de réparation du système. Les valeurs des paramètres sont résumées dans le tableau 5.3 ci-dessous.

Activité	Distribution	Paramètre	Valeur
attaqueVd	Exponentielle	Taux	0.01
attaqueVp	Exponentielle	Taux	0.05
attaqueVc	Exponentielle	Taux	0.01
correctionVul	Normale	Moyenne	30
		Variance	0.5
correctionExp	Normale	Moyenne	7.0
		Variance	0.5
correctionC	Normale	Moyenne	0.6
		Variance	0.5
reparation	Normale	Moyenne	1.0
		Variance	0.5

Tableau 5.3 : Paramétrage du modèle de la vulnérabilité de degré de gravité C2 respectant le scénario de découverte d'origine malveillante

2.2 Résultats des simulations

Nous exécutons notre modèle étendu en mesurant la probabilité de présence de jeton dans chacune des places modélisant les états du système, ainsi que dans les places partagées *C2* et *C3*. Dans cette partie, nous présentons ces résultats. L'instant O de chaque courbe représente l'instant de découverte de la vulnérabilité.

2.2.1 Evolution de l'état du système dans le modèle de processus d'exploitation de vulnérabilité de degré de gravité *C3*

Dans cette partie, nous présentons l'évolution du système dans le modèle tenant compte de la vulnérabilité de degré C3. Nous commençons par l'analyse de cette vulnérabilité car son degré de gravité étant le plus fort, la compromission de cette vulnérabilité est celle qui aura le plus de conséquences sur l'ensemble de l'état du système.

La figure 5.5 présente l'évolution de la probabilité de présence d'un jeton dans la place *Vulnerable*. Cette probabilité décroît fortement jusqu'au seuil d'environ 65% qui correspond à la probabilité d'apparition d'un kit d'exploitation. Cette probabilité reste par la suite décroissante avec une pente très faible durant une longue période entre les jours 10 et 30. La faible décroissance est due aux faibles probabilités d'exploitation et de correction de la vulnérabilité. Cependant, on peut voir que la pente devient plus forte à partir du 30^{e} jour, ce qui traduit l'événement d'application du correctif alors que le système est dans l'état vulnérable.

La figure 5.6 présente l'évolution de la probabilité de présence d'un jeton dans la place *Expose*. Nous retrouvons l'allure de la courbe présentée dans le chapitre 4.

La courbe de la figure 5.7 présente l'évolution de la probabilité de présence d'un jeton dans la place *Compromis*. Nous voyons que cette courbe croit strictement avec une asymptote à environ 35%, ce qui correspond à la probabilité d'existence d'un kit d'exploitation. Cependant, nous pouvons voir que la courbe ne décroît jamais. Cela est dû au blocage du système du au degré de gravité C3 de la vulnérabilité.

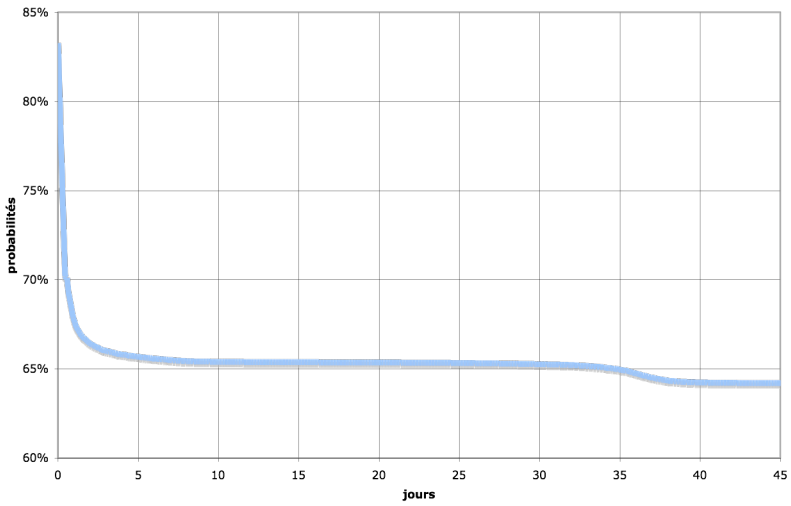


Figure 5.5 : Evolution de la probabilité de présence d'un jeton dans la place *Vulnerable*

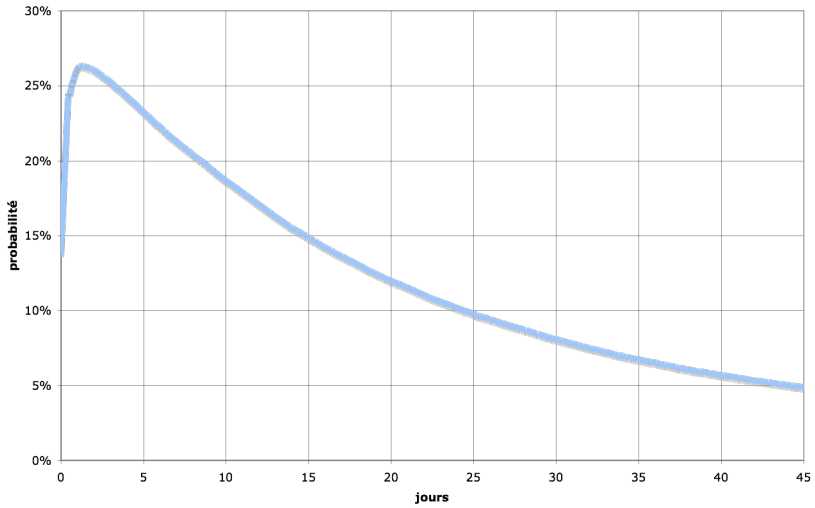


Figure 5.6 : Evolution de la probabilité de présence d'un jeton dans la place *Expose*

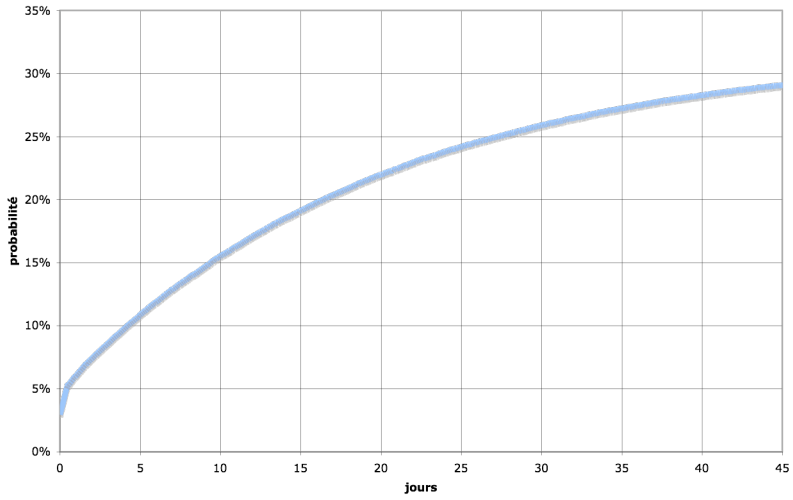


Figure 5.7 : Evolution de la probabilité de présence d'un jeton dans la place *Compromis* – PPC(t)

La figure 5.8 représente l'évolution de la probabilité que le système soit dans l'état sûr vis-à-vis de la vulnérabilité. Nous voyons que la probabilité ne dépasse pas 2% qui est la probabilité d'existence d'un correctif dans ce scénario. Cependant, on remarque la croissance en paliers de la courbe. Ceci est du aux phases successives pendant lesquelles l'administrateur est susceptible d'appliquer le correctif : lorsque le système est dans l'état vulnérable ou l'état exposé. La croissance la plus importante est due au phénomène d'application du correctif durant la phase où le système est vulnérable, la durée moyenne d'application du correctif étant de 30 jours. Cependant, ces deux phénomènes d'application

de correctif de suffisent pas à expliquer l'allure de la courbe entre les jours 0 et 7 qui ne correspond pas à l'allure de la courbe telle que nous l'avions étudié dans le chapitre précédent. Nous allons expliquer ces changements de comportements grâce à l'analyse du sous-modèle de la seconde vulnérabilité dans la partie suivante.

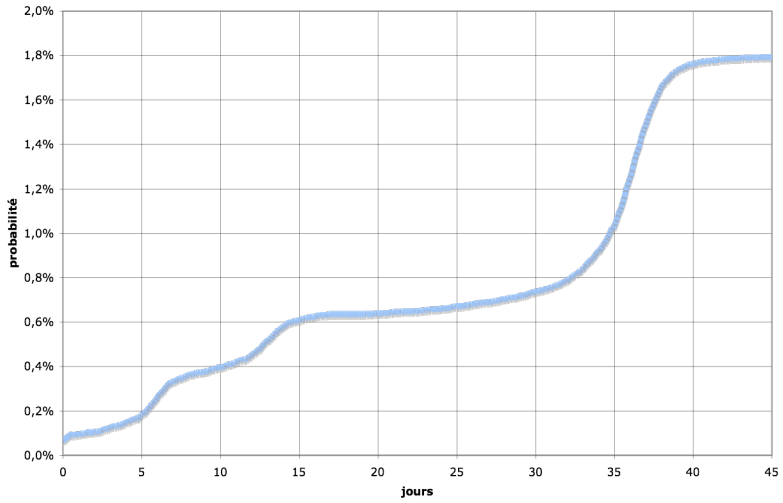


Figure 5.8 : Evolution de la probabilité de présence d'un jeton dans la place *Sur* – PS(t)

2.2.2 Evolution de l'état du système dans le modèle de processus d'exploitation de vulnérabilité de degré de gravité *C2*

Dans cette partie, nous détaillons l'évolution de l'état du système pour la vulnérabilité de degré de gravité C2. Le processus d'exploitation de cette vulnérabilité est associé au scénario de découverte malveillante. Nous avons vu dans le chapitre précédent que l'apparition du kit d'exploitation se faisait très rapidement dès lors que la vulnérabilité était découverte. C'est effectivement le phénomène que nous observons sur la courbe de la figure 5.9 qui montre l'évolution de la probabilité de présence du jeton dans la place *Vulnerable*. Il est normal que cet événement ne soit pas perturbé par les conséquences du processus d'exploitation de la vulnérabilité de degré de gravité C3 car il ne s'agit pas d'une action humaine sur le système mais d'une modification de l'environnement. De plus, l'administrateur ayant un temps d'application du correctif assez long, il est à noter que cet événement intervient de manière peu significative dans cette simulation.

La figure 5.10 montre la probabilité de présence d'un jeton dans la place *Expose*. Nous voyons que cette courbe croit très vite au début avant de décroître moins brutalement, comme nous l'avions vu dans les simulations présentées dans le chapitre 4. Cependant, il est à noter que la probabilité converge vers une valeur constante. Ce comportement est dû à l'augmentation de la probabilité de compromission du système par l'exploitation de la vulnérabilité de degré C3 qui provoque un blocage total du système.

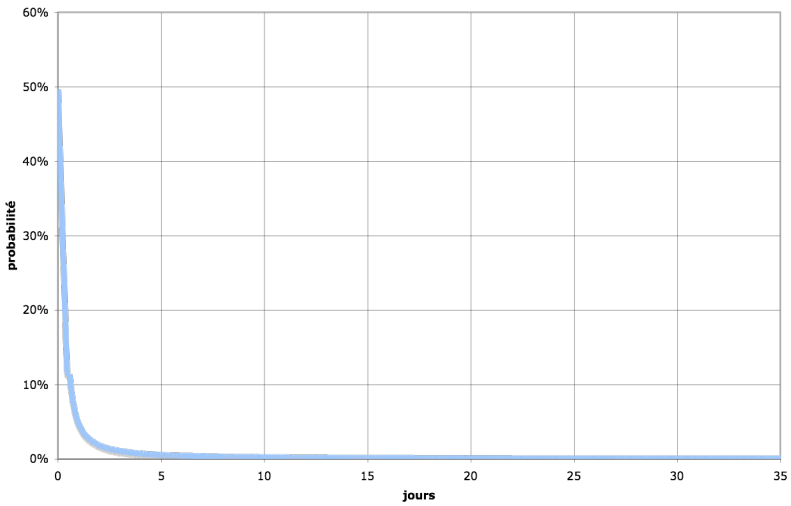


Figure 5.9 : Evolution de la probabilité de présence d'un jeton dans la place *Vulnerable*

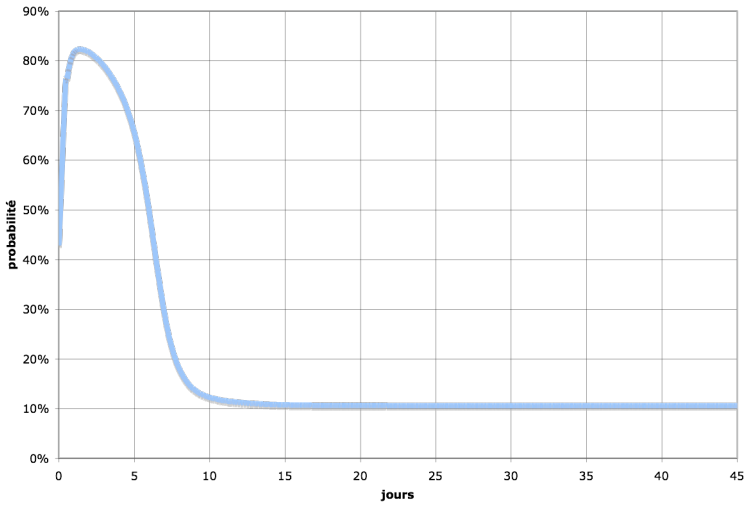


Figure 5.10 : Evolution de la probabilité de présence d'un jeton dans la place *Expose*

Ce comportement est confirmé par les courbes montrant l'évolution des probabilités d'être dans les états compromis, non exposé et sûr, comme cela est montré dans les figures 5.11, 5.12 et 5.13. L'étude d'une seule place aurait suffi à mettre en évidence cette propriété de notre approche. Cependant, il nous a paru intéressant de faire apparaître l'évolution de tous les états du système afin de rester cohérent dans la présentation de nos résultats. De plus, la courbe 5.11, qui montre l'évolution de la probabilité pour le système d'être dans un état compromis vis-à-vis de la vulnérabilité de degré de gravité C2, nous permet également de justifier la pente plus faible de la probabilité d'application du correctif de la vulnérabilité de degré de gravité C3, montrée dans la courbe 5.8. En effet, alors que nous

avions pu identifier les phénomènes d'application du correctif pour la vulnérabilité C3 aux alentours des jours 7 et 30, nous n'avions pas pu donné d'explication pour la tendance de la courbe entre les jours 0 et 5. Ceci s'explique donc par les dépendances entre processus d'exploitation de vulnérabilités.

Notons qu'il est inutile d'étudier les probabilités de présence de jetons des places *C2* et *C3* car elles sont identiques à celles relevées pour les deux places *Compromis* des sous-modèles.

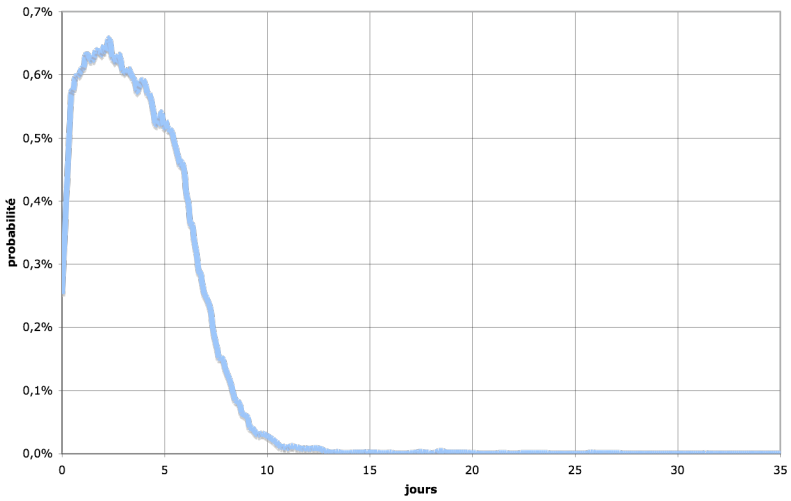


Figure 5.11 : Evolution de la probabilité de présence d'un jeton dans la place *Compromis* – PPC(t)

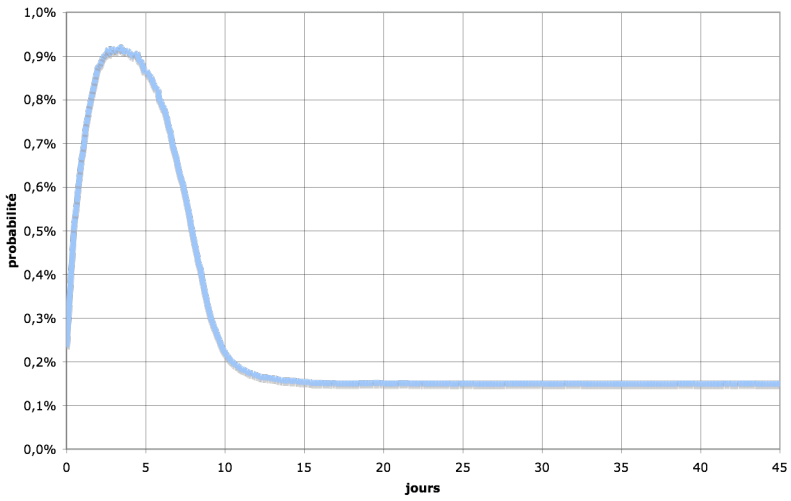


Figure 5.12 : Evolution de la probabilité de présence d'un jeton dans la place *NExpose*

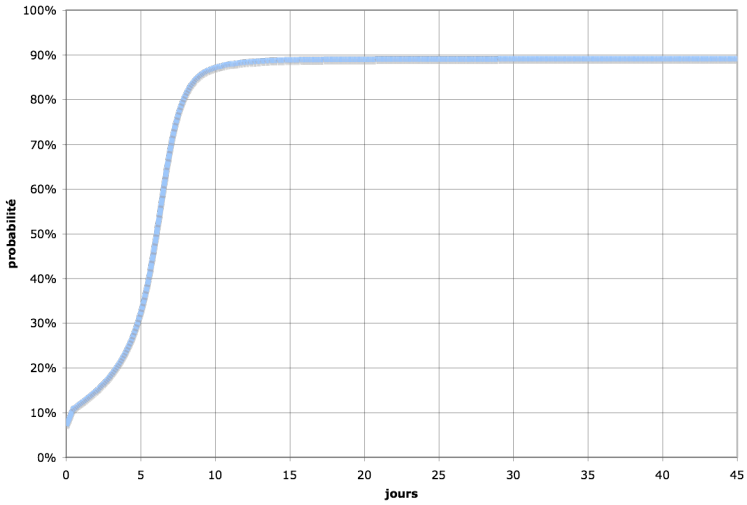


Figure 5.13 : Evolution de la probabilité de présence d'un jeton dans la place *Sur* – PS(t)

3 Conclusion

Dans ce chapitre, nous avons vu comment étendre le principe de notre modélisation pour pouvoir tenir compte de plusieurs vulnérabilités et expérimenté cela en considérant deux vulnérabilités. Cela fonctionne en incluant plusieurs modèles de processus d'exploitation de vulnérabilités dans ce que nous avons appelé un modèle composé. Cette approche utilise une propriété des SAN que nous n'avions pas utilisée jusqu'alors, le *JOIN*, que nous avons présenté ici. Par la suite, nous avons simulé notre modèle à plusieurs vulnérabilités pour valider notre approche.

Nous avons également identifié, dans ce chapitre, plusieurs dépendances entre processus d'exploitation de vulnérabilités que nous n'avons pas modélisées. Ce travail fait partie de nos perspectives. Cette perspective et celles que nous avons évoquées dans ce mémoire sont présentées dans la conclusion générale. Nous y présentons également un résumé de notre approche et un rappel de nos contributions.

Conclusion

Dans ce mémoire, nous avons présenté nos travaux de thèse sur la mise au point d'un processus de mesure quantitative de la sécurité. Après avoir étudié les travaux existants, il est apparu que l'environnement du système était une composante particulièrement importante et qu'il était intéressant d'intégrer l'environnement du système et son évolution dans une approche pour l'évaluation de la sécurité.

Notre objectif était d'élaborer un processus d'évaluation qui pouvait produire des mesures qui soient quantitatives selon deux points de vue complémentaires : la mesure de la distance à un objectif et la mesure de risque considérant une situation donnée. Nous avons gardé cette idée à l'esprit durant l'élaboration de notre modèle. Cependant, nous avons choisi des mesures qui évaluent des risques.

Nous avons choisi d'étudier le processus d'attaque par l'exploitation d'une vulnérabilité. Une première étape de cette approche a donc été l'identification et la caractérisation des facteurs environnementaux importants qui interviennent dans ce processus d'exploitation. Cette étude a mis en évidence trois facteurs environnementaux ayant une influence importante sur le processus d'exploitation d'une vulnérabilité :

- le cycle de vie de la vulnérabilité, comportant les étapes de découverte, publication de la vulnérabilité et publication du correctif ;
- le comportement de la population des attaquants, caractérisé par l'événement de mise au point du kit d'exploitation et par le phénomène d'exploitation de la vulnérabilité ;
- le comportement de l'administrateur, qui est susceptible d'appliquer les correctifs plus ou moins rapidement.

Les dépendances entre ces facteurs environnementaux ont été étudiées et cette étude de l'environnement du système nous a permis de distinguer deux scénarios, l'origine du phénomène de découverte de la vulnérabilité pouvant être malveillant ou non malveillant. Cette étude préliminaire a permis l'élaboration de notre modèle.

Nous avons mis au point deux modèles, correspondant aux deux scénarios identifiés. Le comportement des facteurs environnementaux, ainsi que l'état du système vis-à-vis du phénomène d'exploitation de la vulnérabilité sont pris en considération. Depuis un tel modèle, il est possible d'évaluer les mesures que nous avions définies. Ces contributions ont fait l'objet de plusieurs publications : [VAC 08 ; VAC 09a].

Nous avons validé notre démarche par de premières expérimentations. Nous avons choisi une vulnérabilité connue qui a suscité plusieurs études : la vulnérabilité à l'origine du ver Slammer. Mais la quantification des événements des facteurs environnementaux nécessitant plus de précision et nécessitait d'être rendue générique, nous avons cherché à quantifier les événements de publication de la vulnérabilité, la publication du correctif ainsi que l'apparition du kit d'exploitation. Pour cela, nous avons sélectionné la base de données de vulnérabilités OSVDB et avons analysé 52000 vulnérabilités afin de caractériser quantitativement les probabilités d'occurrence de ces trois événements, dans le contexte des deux scénarios étudiés. Cette étude a mis en évidence deux informations importantes. D'une part, nous avons déterminé les distributions de probabilité permettant de caractériser les événements de publication de la vulnérabilité, publication du correctif et apparition du kit d'exploitation [VAC 09b]. D'autre part, nous avons découvert que les phénomènes de publication du correctif et d'apparition du kit d'exploitation ne sont pas des événements systématiques. A partir de ces nouvelles informations, de nouveaux résultats ont été produits.

Enfin, dans une dernière étape, nous avons élaboré une extension de notre modélisation afin de pouvoir prendre en considération plusieurs vulnérabilités en parallèle.

Notre approche étudie l'environnement du système et son influence sur le processus d'exploitation de vulnérabilités. Ces travaux ont pour but de produire des mesures. Bien que ce mémoire présente les premiers travaux effectués pour notre approche, il semble important de présenter les perspectives que nous envisageons afin de compléter notre approche et la rendre plus pertinente.

Tout d'abord, comme nous l'avons fait pour les événements de publication de la vulnérabilité, de publication du correctif et de la mise au point du kit d'exploitation, il est important de caractériser les autres événements relatifs à l'influence de l'environnement. Les événements restant à caractériser sont le phénomène d'attaque par la population d'attaquants et le phénomène de découverte de la vulnérabilité. Ces deux événements présentent la même difficulté dans le processus de caractérisation, à savoir le manque de données disponibles.

Ensuite, nous avions produit une modélisation très générique, en caractérisant les événements à partir d'un ensemble de vulnérabilités très large. Nous avons évoqué dans notre mémoire que la caractérisation utilisant les catégories de vulnérabilités fournies par la base de données n'avait pas permis d'associer une catégorie de vulnérabilités avec un type de comportement du cycle de vie ou de la population d'attaquants. Une perspective que nous envisageons est d'étudier cet ensemble de vulnérabilités en adoptant la démarche inverse : catégoriser les vulnérabilités ayant le même profil en terme de cycle de vie et d'événement d'apparition du kit d'exploitation et examiner si les catégories ont un sens en terme de type d'attaque ou de logiciel source. Ce travail permettrait d'affiner encore la modélisation et d'obtenir des mesures plus précises. Cela permettrait également de configurer plus facilement les modèles composés destinés à la simulation de plusieurs vulnérabilités en parallèle.

Nous avons également présenté, dans le dernier chapitre de ce mémoire, une extension de notre modélisation pouvant considérer plusieurs processus d'exploitation de vulnérabilités en parallèle. Nous avons évoqué des dépendances entre les modèles que nous n'avons pu formaliser et intégrer à notre modélisation, faute de temps. L'ajout de ces contraintes ouvre une perspective de nos travaux. De plus, cet aspect de notre travail aborde le problème de la caractérisation du phénomène d'apparition de la vulnérabilité elle-même.

Enfin, cette approche considère les vulnérabilités créées durant la phase de développement. Nous avons volontairement laissé de coté les vulnérabilités de configuration car celles-ci ne possèdent pas de cycle de vie comme celui que nous avons étudié pour les vulnérabilités de développement. Cependant, les attaques exploitant les vulnérabilités de configuration existent et sont également la source de la compromission de nombreux systèmes. Le premier exemple est le principe des attaques par dictionnaire qui tentent d'établir une connexion essayant plusieurs couples de noms d'utilisateurs et de mots de passe. Une de nos perspectives est d'étendre encore notre approche afin de pourvoir évaluer les risques encourus également avec cette catégorie de vulnérabilités.

Bibliographie

[ABG 04] N. Abghour, *Schéma d'autorisation pour applications réparties sur Internet*, Thèse de doctorat, Institut National Polytechnique de Toulouse, 2004 (Rapport LAAS n° 04327).

[ALA 05] Alata, E. et Dacier, M. et Deswarte, Y. et Kaâniche, M. et Kortchinsky, K. and Nicomette V. and Pham, V.H. and Pouget F. *Leurré.com: retour d'expérience sur plusieurs mois d'utilisation d'un pot de miel distribué mondialement*; Conférence invitée, Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'05), Rennes (France), 1-3 Juin 2005, pp.281-292

[ALA 06a] Alata, E. and Dacier, M. and Deswarte, Y. and Kaâniche, M. and Kortchinsky, K. and Nicomette, V. and Pham, V.H. and Pouget, F. *Collection and analysis of attack data based on honeypots deployed on the Internet*, Quality of Protection – Security Measurements and Metrics, pp 79-91, 2006.

[ALA 06b] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, M. Herb, *Lessons learned from the deployment of a high-interaction honeypot,* Proceedings of the 6th European Dependable Computing Conference (EDCC-6), Coimbra, Portugal, Oct 18-20, 2006, IEEE Computer Society, pp 39-44.

[ALA 07] E. Alata, *Observation, Caractérisation et modélisation de processus d'attaques sur Internet*, Thèse de Doctorat, Rapport LAAS n°07805, 2007.

[ALL 07] X. Allamigeon, C. Hymans, Analyse statique par interprétation abstraite, Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC'07), 2007.

[AND 80] J. P. Anderson, *Computer Security Threat – Monitoring and Surveillance*, J. P. Anderson CO. Technical report, Fort Washington, Pennsylvania, 1980.

[ARB 00] W. A. Arbaugh, W. L. Fithen, J. McHugh, *Windows of Vulnerability : a case study analysis*, Computer, vol. 33, no. 12, pp. 52-59, 2000.

[ARO 04] Arora A. and,Krishnan R. and Nandkumar A. and Yang Y., *Impact of vulnerability disclosure and Patch availability – an empirical analysis,* in Third Workshop on the Economics of Information Security, 2004.

[ATZ 06] A. Atzeni, A. Lioy, *Why to adopt a security metric ? A brief survey*, Quality of protection – Security measurements and metrics, 2006, p1-12.

[AVI 04] A. Avizienis, J-C. Laprie, B. Randell, C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing*, IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, January-March 2004, pp 11-34.

[BAL 06] D. Balzarotti, M. Monga, S. Sicari, *Assessing the risk of using vulnerable components*, Quality of protection – Security measurements and metrics, 2006, p65-77.

[BEO 93] C. Béounes, M. Aguéra, J. Arlat, S. Bachmann, C. Bourdeau, J-E. Doucet, K. Kanoun, J-C. Laprie, S. Metge, J. M. D; Souza, D; Powell et P. Speisser, *Surf-2: a program for dependability evaluation of complex hardware and software systems*, in 23rd IEEE Int. Symposium of Fault Tolerant Computing, Toulouse, France, pp 668-673, 1993.

[BUG] http://www.securityfocus.org

[CC 96] *Common Criteria for Information Technology Security Evaluation, Part1: Introduction and General Model*, CCEB-96/011, 1996.

[CER 05] *The development of meaningful hacker taxonomy: a two dimensional approach*, CERIAS Technical Report 2005-43, 2005.

[CLA 01] Clark, G., Courtney, T., Daly, D., Deavours, D., Derisavi, S., Doyle, J. M., Sanders, W. H., and Webster, P. 2001. The Möbius Modeling Tool. In Proceedings of the 9th international Workshop on Petri Nets and Performance Models (Pnpm'01) (September 11 - 14, 2001).

[CLU 03] Club de la Sécurité des Systèmes d'Information Français, *ISO17799:2000 – Une présentation générale*, Dossier technique, Février 2003.

[CNI] "Commission Nationale de l'Informatique et des Libertés" : http://www.cnil.fr

[COU 04] Courtney, T., Daly, D., Derisavi, S., Gaonkar, S., Griffith, M., Lam, V., and Sanders, W. H. 2004. The Möbius Modeling Environment: Recent Developments. In Proceedings of the Quantitative Evaluation of Systems, First international Conference (September 27 - 30, 2004).

[CUP 02] F. Cuppens et A. Miege : Alert correlation in a cooperative intrusion detection framework. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 202–215, Washington, DC, USA, 2002. IEEE Computer Society.

[CUP 06] F. Cuppens, N. Cuppens, *Les modèles de sécurité*, chapitre de "Sécurité des systèmes d'information, (Traité IC2) sous la direction de Y. Deswarte et L. Mé, pp 13-48, série Réseaux et télécoms, Hermès-Lavoisier, 2006.

[CTC 93] *The Canadian Trusted Computer Product Evaluation Criteria*, version 3.0e, Canadian System Security Center, Communication Security Establishment of Canada, 1993.

[CVE] "Common Vulnerability Exposure" : http://cve.mitre.org

[DAC 94] M. Dacier, *Vers une évaluation quantitative de la sécurité*, Thèse de doctorat, Rapport LAAS n° 94488, 1994.

[DAC 96] M. Dacier, Y. Deswarte & M. Kaâniche. *Quantitative assessment of operational security : models and tools*, 12th IFIP Information System Security Conference (S. K. Katsikas, D. Gritzalis, Eds.), Samos, Greece, May 21-24, Chapman & Hall, 1996 (Rapport LAAS n° 96493)

[DAC 06] M. Dacier, L. Mé, Y. Deswarte, *Sécurité des systèmes d'information* (Traité IC2, série Réseaux et télécoms), *chapitre Détection d'intrusions : état de l'art, faiblesses et problèmes ouverts*. Lavoisier, 2006.

[DES 91] Y. Deswarte, L. Blain et J.C. Fabre : *Intrusion Tolerance in Distributed Computing Systems*, Proceedings of the 1991 IEEE Symposium on Security and Privacy, pp. 110–121, Oakland, CA, USA, 1991.

[DOD 85] *U.S. Department of Defence Trusted Computer Security Evaluation Criteria*, 5200-28-STD, 1985.

[DUM 98] W. Du, A. P. Mathur, *Categorization of Software Errors that led to Security Breaches*, In 21st National Information Systems Security Conference, 1998.

[ERN 05] Ernst & Young, *Global Information Security Survey*, 2005.

[FAR 90] D. Farmer, E.H. Spafford, *The COPS security checker system,* USENIX Summer, pp. 165–170, 1990.

[FC 92] *Federal Criteria for Information Technology Security*, Draft, Volume I & II, National Institute of Standards and Technology (NIST) and National Security Agency (NSA), 1992.

[FIS 03] N. Fischbach, "Le cycle de vie d'une vulnérabilité", 2003.

[FRE 06] S. Frei, M. May, U. Fiedler, B. Plattner, *Large scale vulnerability analysis*, Proceedings of the 2006 SIGCOMM workshop on Large-Scale attack

defense, Pisa, Italy, ACM, 2006, pp 131-138.

[FRE 09] Frei, S. *Security Econometrics – The Dynamics of (In)Security*, Doctorate Thesis, March 28, 2009.

[GAD 08] M. Gadelrab, *Evaluation of Intrusion Detection Systems*, These de doctorat, 2008.

[HAU 03] E. Haugh, M. Bishop, *Testing C programs for buffer overflow vulnerabilities,* Proceedings of the Network and Distributed System, Security Symposium (NDSS'03), San Diego, CA, USA, 2003. The Internet Society.

[HET 89] H. W. Hetcote, *Three basic Epidemiological Models*, Biomathematics vol. 18, pp 119-144, Springer Berlin, 1989.

[IRV 99] C. Irvine, T. Levin, *Toward a Taxonomy and Costing Method for Security Services*, 15th Computer Security Applications Conference, 6-10 Dec, 1999, pp 183-188.

[ISO 00] *ISO/IEC 17799:2000 – Code of practice for Information Security Management*.

[ISO 08] "An Introduction to ISO 27001, ISO 27002....ISO 27008" : http://www.27000.org

[ITS 91] *Information Technology Security Evaluation Criteria*, European Communities, 1991.

[ITS 93] *Information Technology Security Evaluation Manual*, European Communities, 1993.

[JAQ 07] A. Jaquith, Security Metrics, Replacing Fear, Uncertainty and Doubt, Addison-Wesley Editions, 2007.

[JHA 02] S. Jha, O. Sheyner and J. Wing, *Two formal analyses of attack graphs*, Computer Security Foundation Workshop, 2002.

[JON 07] J. R. Jones, *Estimating Software Vulnerabilities*, IEEE Security and Privacy, Jul-Aug 2007, vol. 5, issue 4, pp 28-32.

[JUM 08] Jumratjaroenvanit, A. and Teng-amnuay Y. *Probability of Attack based on System Vulnerability Life Cycle*, isecs, pp.531-535, 2008 International Symposium on Electronic Commerce and Security, 2008.

[KEM 60] J. G. Kemeny, J. L. Snell. Finite Markov Chains. D.Van Nostrand Co., Inc., Princeton, New Jersey, USA, 1960.

[KIM 04] J. Kim, S. Radhakrishnan, S. K. Dhall, *Measurement d Analysis of Worm Propagation on Internet Network Topology*, International Conference on Computer Communications and Networks, 2004.

[KUH 07] Kuhl, M.E. and Kistner, J. and Constantini, K. and Sudit, M. *Cyber attack modeling and simulation for network security analysis,* Proceedings of the 2007 Winter Simulation Conference, 2007.

[LAP 04] J-C. Laprie, *Sûreté de Fonctionnement des Systèmes : Concepts de base et Terminologie*, REE – Revue de l'Electricité et de l'Electronique – no. 11, décembre 2004, pp 95-105.

[LAP 08] J-C Laprie, *From Dependability to Resilience*, Fast Abstract of the 38th International Conference of Dependable System Network, Anchorage, Alaska, USA, June 24-27, 2008.

[LAP 95] J-C. Laprie,J. Arlat, J-P. Blanquart, A. Costes, Y. Crouzet, Y.Deswartes, J-C. Fabre, H. Guillermain, M. Kaâniche, K. Kanoun, C. Mazet, D. Powell, C. Rabéjac, P. Thévenod, *Guide de la sûreté de fonctionnement*, 2e édition, Cépaduès, 1995.

[LAR 01] D. Larochelle, D. Evans, Statically detecting likely buffer overflow Vulnerabilities, Proceedings of the 10th conference on USENIX Security Symposium (SSYM'01), pp. 14–14, Berkeley, CA, USA, 2001.

[LAR 02] Dictionnaire Larousse de la langue française, 2002.

[LAU 02] A.P. Laudicina, *Nessus – a powerful, free remote security scanner*, SysAdmin : The Journal for UNIX Systems Administrators, vol. 11, num. 5, pp. 24, 26, 28–30, 2002.

[LEU] "Leurrecom.org honeypot project" : http://www.leurrecom.org/

[LIT 93] B. Littlewood, S. Broklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, D. Gollmann, *Towards Operational Measures of Computer Security*, 1993.

[LIU 05] P. Liu, W Zang, *Incentive-Based Modeling and Inference of Attaker Intent, Objectives and Strategies*, ACM Transactions on Information and System Security (TISSEC), Vol. 8, Issue 1, p78-118, February 2005.

[MAD 02] B. B. Madan, K. S. Trivedi, Security modeling and quantification of intrusion tolerant systems, ISSRE, 2002.

[MAD 02a] B. B. Madan, K. Goseva-Popstosanova, K. Vaidyanathan, K. S. Trivedi, *Modeling ans Quantification of Security Attributes of Software Systems*, Proceedings of Dependable Syystems and Networks, 2002.

[MAF 01] *Malicious and Accidental Fault Tolerance in Internet Applications : Towards a Taxonomy of Intrusion Detection Systems and Attacks*, MAFTIA Project Delivrable D3, Version 1.01, Septembre 2001.

[MAF 03] *Malicious and Aaccidental Fault Tolerance in Internet Applications : conceptual model and architecture*, MAFTIA Project Delivrable D21, 2003 .

[MAT] "Easy Fit – Mathwave tool" : http://www.mathwave.com/products/easyfit.html

[MAR 95] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis. *Modeling with Generalized Stochastic Petri Nets*. John Wiley & Sons Ltd., 1995.

[MCQ 06] M.A. McQueen, W.F. Boyer, M.A. Flynn, G.A. Beitel, *Time-to-compromise Model for cyber risk reduction estimation,* Quality of protection – Security measurements and metrics, 2006, p49-64.

[MEI 07] Meil, P. and Scarfone, K. and Romanovsky S. *A Complete Guide to the Common Vulnerability Scoring System – Versions 2.0.* http://www.first.org/cvss, June 2007

[MEM 01] L. Mé, Z. Marrakchi, C. Michel, H. Debar et F. Cuppens, *La détection d'intrusions : les outils doivent coopérer,* La Revue de l'Electricité et de l'Electronique, n° 5, pp. 50–55, 2001.

[MEM 01a] L. Mé, C. Michel, *Intrusion detection : A bibliography,* Rapport technique SSIR-2001-01, Supélec, Rennes, France, 2001.

[MET] "The Metasploit Project" : http://www.metasploit.com/

[MIC 02] "Microsoft Security Bulletin MS02-039." : http://www.microsoft.com/technet/security/bulletin/MS02-039.mspx

[MIR 04] "Misra C" : www.misra-c2.com

[MIR 07] Mirkovic, J., Hussain, A., Willson, B., Fahmy, S., Yao, W., Reiher, P., Schwab, S., and Thomas, R, *When is service really denied?: a user-centric dos metric*, Proceedings of the 2007 ACM SIGMETRICS international Conference on Measurement and Modeling of Computer Systems, San Diego, California, USA, June 12-16, 2007), pp 357-358.

[MOO 03] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, et N. Weaver, "Inside the Slammer worm," *Security & Privacy, IEEE*, vol. 1, 2003, pp. 33-39.

[NGK 08] B-Y. Ng, A. Kankanhalli, Y. Xu, *Studying user's computer security behaviot : a health belief perspective*, Journal of Decision Support Systems, vol. 46, March 2009, pp 815-825.

[NOE 04] Noel, S. and Jajodia, S. 2004. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining For Computer Security* (Washington DC, USA, October 29 - 29, 2004).

[NRC 02] National Research Council, *Cyber Security Today and Tomorrow*, 2002.

[NVD] "Nationale Vulnerability Database" : http://nvd.nist.gov/

[OKA 05] H. Okamura, H. Kobayashi, T. Dohi, *Markovian Modeling and Analysis of Internet Worm Propagation*, ISSRE 2005.

[ORT 98] R. Ortalo. *Evaluation quantitative de la sécurité des systèmes d'information*, Thèse de doctorat, Institut National Polytechnique de Toulouse, 1998 (Rapport LAAS n° 98164).

[ORT 99] R. Ortalo, Y. Deswarte, M. Kaaniche, *Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security*. IEEE Transactions on Software Engineering, vol. 25, num. 5, pp. 633– 650, 1999, Los Alamitos, CA, USA.

[OSV] "The Open Source Vulnerability Database" : http://osvdb.org

[PAY 01] S. C. Payne, *A Guide to Security Metrics*, SANS Security Essentials GSEC Pratical Assignment Version 1.2e, July 2001.

[RAM 07] E. Ramirez-Silva, M. Dacier, *Empirical study of the impact of metasploit-related attacks in 4 years of attack traces*, ASIAN'07, 12th Annual Asian Computing Science Conference Focusing on Computer and Network Security, December 9-11, 2007, Doha, Qatar, pp 198-211

[RES 05] E. Rescorla, *Is fiding security holes a good idea?*, IEEE Security and Privacy, Vol3, issue 1, Jan-Feb 2005, pp 14-31.

[RES 07] Banatre, M. and Patarizca, A. and Van Moorsel, A. and Palanque, P. and Strigini, L. From Resilience Building to Resilience Scaling Technologies : Directions, delivrable D13, Excellence Network ReSIST, www.resist-noe.org, November 2007.

[RIC 02] G. Rice and J. Davis, *a genealogical approach to analysing post-mortem denial of service attacks*, 2002.

[ROG 06] Rogers, M. K. *A two-dimensional circumplex approach to the development of a hacker taxonomy*, Digital Investigation, Volume 3, Issue 2, June 2006, Pages 97-102.

[SAI 05] A. Saidane, *Conception et réalisation d'une architecture tolérant les intrusions pour des serveurs internet*, Thèse de Doctorat, Rapport LAAS n° 05147, 2005.

[SAI 09] A.Saidane, Y. Dewarte, V. Nicomette, *The design of a generic intrusion tolerant architecture for Web servers*, IEEE Transactions on Dependable and Secure Computing, vol.6, issue 1, Jan-March 2009, pp 45-58.

[SAL 05] K. Sallhammar, S. J. Knapskog, B. E. Helvik, *Using Stochastic game theory to compute the expected behavior of attackers*, Symposium on applications and the Internet Workshops, 2005.

[SAL 06] K. Sallhammar, B. E. Helvik, S. J. Knapskog, *Towards a Stochastic Model for Integrated Security and Dependability Evaluation*, Proceedings of the First International Conference On Availability, Reliability and Security (ARES'06), 2006.

[SAN 02] Sanders, W. H. and Meyer, J. F. 2002. *Stochastic activity networks: formal definitions and concepts*. In Lectures on Formal Methods and Performance Analysis: First Eef/Euro Summer School on Trends in Computer Science, E. Brinksma, H. Hermanns, and J. Katoen, Eds. Springer Lectures On Formal Methods And Performance Analysis, vol. 2090.

[SCH 99] B. Schenier, Attack trees – Modeling security threats, www.schenier.com.

[SHE 04] O. M. Sheyner. *Scenario graphs and attack graphs*, Thesis of School of Computer Science, Computer Science department, Carnegie Mellon University, Pittsburgh, PA, 2004.

[SEC b] "Secunia" : http://secunia.com

[SEC a] "Security Focus" : http://www.securityfocus.com

[SMV 01] "The SMV System" : http://www.cs.cmu.edu/~modelcheck/smv.html

[SNO 09] "The de facto standard for intrusion detection/prevention" : http://www.snort.org

[STE 59] S. S. Stevens, *Measurement, Psychophysics and Utility*, Chap. 2 of *Measurement: Definitions and Theories*, in C. W. Churchman & P. Ratoosh (Eds.), 1959.

[STU 26] HA Sturges, *The choice of a class interval*, Journal of the American Statistical Association, 1926.

[SWI 01] L.P. Swiler, C. Phillips, D. Ellis, S. Chakerian, *Computer-Attack Graph Generation Tool*, DARPA Information Survivability and Exposition II, 2001.

[TRA 05] Z. Trabelsi, H. Ly, La sécurité sur Internet, Hermès Sciences Publications, ed. Lavoisier, 2005.

[VAC 08] G. Vache, *Toward Information System Security Metrics*, Proceedings of EDCC-7 (European Dependable Computing Conference), vol. 2, pp 41-44, Kaunas, Lithuania, 7-9 May 2008.

[VAC 09a] G. Vache, *Environment characterization and system modeling approach for the quantitative evaluation of security*, Procedings of SAFECOMP 2009 (28th International Conference on Computer Safety, Reliability and Security), pp 89-102, Hamburg, Germany, 15-18 September 2009.

[VAC 09b] G. Vache, *Vulnerability analysis for a quantitative security evaluation*, Proceedings of METRISEC (International Workshop on Security Measurements and Metrics), Lake Buena Vista, Florida, USA, October 14, 2009.

[VER 06] P. Verissimo, N. F. Neves, C. Cachin, J. Poritz, D. Powell, Y. Deswarte, R. Stroud, I. Welch, *Intrusion-Tolerant Middleware : The Road to Automatic Security,* IEEE Security and Privacy, vol. 4, num. 4, pp. 54–62, 2006.- 16 May.

[VIR] "Viruslist.com : Information abour Viruses, Hackers and Spam" : http://www.viruslist.com

[WAL 06] M. Walter, C. Trinitis, *Quantifying the secirity of Composed System*, Proceedings of the Workshop on Dependability of the Distributed Systems, pp 1026-1033, 2006.

[WAN 97] C. Wang, W. A. Wulf, *A framework for security measurement*, Proceedings of the National Information Systems Security Conference, Baltimore, MD, pp. 522-533, October 1997.

[WAN 05] A. J. A. Wang, *Information Security Models and Metrics*, Proceedings of the 43rd ACM Southeast Conference, vol. 2, pp 178-184, Kennesaw, GA, USA, March 18-20, 2005.

[WAS 07] Wash, R. and MacKie-Mason, J. K. 2007. *Security when people matter: structuring incentives for user behavior*. In *Proceedings of the Ninth international Conference on Electronic Commerce* (Minneapolis, MN, USA, August 19 - 22, 2007).

[WHI 03] G. Whitson, *Computer Security : Theory, Processes and Management*, Consortium for Computing Sciences in Colleges, JCSC 18, 2003.

[ZUS 91] H. Zuse, *Software complexity, Measures and methods*, Walter de Gruyter Editions, 1991.

Cette thèse présente une nouvelle approche pour l'évaluation quantitative de la sécurité des systèmes informatiques. L'objectif de ces travaux est de définir et d'évaluer plusieurs mesures quantitatives. Ces mesures sont des mesures probabilistes visant à quantifier les influences de l'environnement sur un système informatique en présence de vulnérabilités.
Dans un premier temps, nous avons identifié les trois facteurs ayant une influence importante sur l'état du système : 1) le cycle de vie de la vulnérabilité, 2) le comportement de la population des attaquants et 3) le comportement de l'administrateur du système. Nous avons étudié ces trois facteurs et leurs interdépendances et distingué ainsi deux scénarios principaux, basés sur la nature de la découverte de la vulnérabilité, malveillante ou non. Cette étape nous a permis d'identifier les états possibles du système en considérant le processus d'exploitation de la vulnérabilité et de définir quatre mesures relatives à l'état du système qui peut être vulnérable, exposé, compromis, corrigé ou sûr.
Afin d'évaluer ces mesures, nous avons modélisé ce processus de compromission. Par la suite, nous avons caractérisé de manière quantitative les événements du cycle de vie de la vulnérabilité à partir de données réelles issues d'une base de données de vulnérabilités pour paramétrer nos modèles de manière réaliste. La simulation de ces modèles a permis d'obtenir les valeurs des mesures définies.
Enfin, nous avons étudié la manière d'étendre le modèle à plusieurs vulnérabilités.
Ainsi, cette approche a permis l'évaluation de mesures quantifiant les influences de différents facteurs sur la sécurité du système.
Mots-clés : sécurité, mesures quantitatives, évaluation, vulnérabilité

This thesis presents a new approach for quantitative security evaluation for computer systems. The main objective of this work is to define and evaluate several quantitative measures. These measures are probabilistic and aim at quantifying the environment influence on the computer system security considering vulnerabilities.
Initially, we identified the three factors that have a high influence on system state: 1) the vulnerability life cycle, 2) the attacker behaviour and 3) the administrator behaviour. We studied these three factors and their interdependencies and distinguished two main scenarios based on nature of vulnerability discovery, i.e. malicious or non malicious. This step allowed us to identify the different states of the system considering the vulnerability exploitation process and to define four measures relating to the states of the system: vulnerable, exposed, compromised, patched and secure.
To evaluate these measures, we modelled the process of system compromising by vulnerability exploitation. Afterwards, we characterized the vulnerability life cycle events quantitatively, using real data from a vulnerability database, in order to assign realistic values to the parameters of the models. The simulation of these models enabled to obtain the values of the four measures we had defined.
Finally, we studied how to extend the modelling to consider several vulnerabilities.
So, this approach allows the evaluation of measures quantifying the influences of several factors on the system security.
Keywords : security, quantitative maesures, evaluation, vulnerability