Action Spécifique
Sécurité logicielle: modèles et vérification
Programme de travail

Hubert Comon-Lundh Frédéric Cuppens Thomas Jensen
Yassine Lakhnech

Novembre 2002

Table des matières

1 Présentation générale

1.1 Introduction

La présente action a pour but premier de réunir les équipes travaillant sur les aspects formels de la sécurité logicielle en France, et qui sont actuellement éparpillées et peu coordonnées. En particulier, il existe une grande variété de modèles spécifiques à des systèmes informatiques tenant compte d'un environnement hostile. Les applications sont aussi variées : carte à puce, détection d'intrusion, protocoles cryptographiques. Il existe enfin une grande variété de techniques de vérification de propriétés de sécurité.

Beaucoup d'équipes venant d'horizons très variés se sont investies dans ces domaines plus ou moins récemment. Elles ont chacune une culture et des compétences propres, mais possèdent en commun l'objectif d'utiliser des méthodes formelles pour garantir (autant que possible) des propriétés de sécurité comme, par exemple, la confidentialité.

Il est important (et urgent) de partager les compétences et les points de vue en matière de sécurité logicielle. Nous espérons bien sûr que ces confrontations permettront de faire émerger de nouvelles idées et de nouvelles coopérations. Nous espérons également que l'action permettra d'accroître la visibilité de nos équipes de recherche, notamment en vue de transferts industriels et d'actions européennes.

1.2 Stucture et objectifs

Notre action est structurée selon trois axes spécifiques :
- La vérification de protocoles cryptographiques
- La vérification de code embarqué
- Les modèles pour la disponibilité et la survivabilité

qui font chacun l'objet d'un programme de recherche plus détaillé dans les sections 2, 4 et 3.

Cependant, l'intérêt majeur de cette action est la fusion de ces trois domaines de recherche. Les représentants de chacun de ces axes se sont réunis le 14 novembre à Cachan. Ils ont exposé les travaux et problèmes dans leur domaine. Il est apparu, comme nous l'avions anticipé, que les questions fondamentales sont les mêmes dans les trois axes, même si les approches sont différentes.

En particulier, il est apparû que tous ceux qui travaillent sur les aspects formels de la sécurité partagent les préoccupations suivantes, que l'on retrouvera dans les sections 2, 3 et 4 :

Quelles propriétés de sécurité ? Il existe un besoin de plus de précision dans les modèles utilisés. En particulier, la définition de logiques permettant de décrire des propriétés sécuritaires est un prérequis à toutes les questions de vérification. Or il apparaît que, dans les trois cas, cette définition n'existe pas ou, plutôt, qu'il y a de multiples définitions contradictoires. Il nous semble qu'il est utile, voire nécessaire de confronter nos points de vue.

Quels sont les mécanismes et fonctions de sécurité censés satisfaire ces propriétés ? Le premier point ci-dessus consiste en bref à préciser un cahier des charges. Ici, il s'agit de savoir quels sont les mécanismes supposés remplir ce cahier des charges. Ce n'est pas si simple de répondre à la question tant les moyens mis en oeuvre sont nombreux et disparates.

Les propriétés sont elles satisfaites ? Le problème, après avoir répondu aux deux premières questions, est celui de l'adéquation des moyens mis en oeuvre et des propriétés qui sont supposées être satisfaites. Est-ce que les objectifs sécuritaires sont atteints ? Si non (ce qui sera le cas général), sous quelles hypothèses supplémentaires sont elles satisfaites ? Peut-on proposer des scenarios d'attaque ? Enfin, les réponses aux deux premières questions permettent aussi d'étudier et de comparer les outils de validation, notamment en précisant leur portée.

1.3 Programme de travail

Nous nous proposons de répondre concrètement aux questions ci-dessus par :

1. La mise en service de sites web permettant d'avoir accès à des travaux de synthèse, des catalogues de propriétés, de logiciels censés les réaliser, d'outils de vérification.
2. La tenue de deux réunions dans l'année qui permettront de coordonner nos activités, échanger des résultats, mettre sur pied des actions internationales, étudier des actions de transfert.

Les programmes plus détaillés sont proposés dans les sections 2, 3 et 4.

1.4 Bénéfice espéré pour la communauté scientifique française

En bref, notre action a une ambition de clarification, qui devra conduire à un accroissement de visibilité, notamment par publication (via des sites web) de synthèses. Ceci devra permettre l'accroissement des coopérations industrielles, qui sont pour l'instant très en dessous de ce qu'elles pourraient être sur les problèmes de sécurité.

Cette clarification devrait permettre aussi de structurer la recherche française dans le domaine ; par exemple, nous envisageons cette action pour un an comme un tremplin pour le montage d'autres actions, nationales ou internationales, plus ciblées, en fonction des thèmes de recherche communs qui auront émergé.

Ce travail commun de tous les participants à un effort de clarification ne peut se faire qu'en réunissant les principaux acteurs de la recherche française dans le domaine des aspects formels de la sécurité. C'est pourquoi cette action comporte un nombre inhabituel de participants. Nous espérons qu'il ressort de la lecture des sections 2,3 et 4 qu'il existe des problématiques transversales et que nous avons la volonté de les dégager ensemble.

1.5 Participants à l'action

Responsables de l'action : H. Comon-Lundh (`Hubert.Comon@lsv.ens-cachan.fr`), F. Cuppens (`cuppens@irit.fr`), T. Jensen (`Thomas.Jensen@irisa.fr`), Y. Lakhnech (`Yassine.Lakhnech@imag.fr`)

Laboratoires concernés : Laboratoire Spécification et Vérification (ENS Cachan), Verimag (Grenoble), Laboratoire d'Informatique de Marseille, Loria (Nancy), IRISA (Rennes), Laboratoire Bordelais de Recherche en Informatique (Bordeaux), projet Lemme (INRIA Sophia), Laboratoire d'Informatique Fondamentale de Lille, Institut de Recherche en Informatique de Toulouse, ONERA (Toulouse), Laboratoire d'Analyse et d' Architecture des Systèmes (Toulouse), Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour.

L'action concerne une équipe de chacun de ces laboratoires ou projets.

Invités industriels : France Telecom R & D, Trusted Logic, Serma Technologie MasterSecurity.

1.6 Moyens demandés au CNRS

Nous demandons 50000 euros pour un an. Ceci permettra d'organiser les deux réunions prévues qui doivent rassembler environ 35 participants sur deux jours chacune (pour environ 20000 euros), de financer les visites d'un laboratoire à l'autre de membres participants (pour environ 9000 euros) ainsi que des missions à l'étranger (pour environ 21000 euros).

2 Les défis de la vérification automatique de protocoles cryptographiques

2.1 Les enjeux

Les protocoles cryptographiques sont utilisés dans de nombreuses applications accomplissant des transactions électroniques. Leur spécificité vient des points suivants :

- On souhaite assurer certaines propriétés sécuritaires, typiquement la confidentialité, tout en utilisant des canaux de communication publics
- Les ressources utilisées par les participants sont supposées très limitées, de manière à ce que la mise en oeuvre des protocoles soit transparente pour les utilisateurs.
- Au contraire, les capacités d'un intrus sont supposées très largement supérieures, dans la mesure où celui-ci peut concentrer ses moyens sur un petit nombre de messages bien choisis.
- Les canaux publics ne sont pas fiables, si bien qu'un intrus a la capacité d'intercepter des messages et d'envoyer des messages truqués.

Beaucoup d'entreprises petites et grandes mettent au point leurs propres protocoles, dans le cadre des applications spécifiques qu'elles développent. Elles tiennent en général jalousement secrets ces protocoles.

Il apparait ainsi que la vérification automatique de propriétés sécuritaires de ces protocoles est un enjeu important dans les années à venir.

2.2 Le contexte

Le problème de la vérification des protocoles cryptographiques n'est pas nouveau : il remonte aux années 1980. Cependant, d'une part son importance n'a fait que croître et d'autre part, ce n'est que depuis 1998 environ que des cadres formels (sémantique) ont été proposés pour effectuer ces vérifications, les travaux précédents se bornant à la recherche d'attaques.

La recherche sur ce sujet a principalement lieu en France, en Grande Bretagne, en Italie et aux Etats Unis. Plusieurs outils de démonstration automatique ont été mis au point ces dernières années. En France on peut citer les groupes de Marseille, de Grenoble, de Nancy et de Cachan, ainsi que B. Blanchet à Paris.

Tous les outils (en France comme ailleurs) présentent de nombreuses faiblesses, que nous résumons dans le paragraphe suivant. Ces difficultés viennent d'abord du fait que la vérification des protocoles cryptographiques est indécidable, même avec des hypothèses très fortes, comme de nombreux articles l'ont montré. En deuxième lieu, il s'est avéré que beaucoup d'attaques sont dûes non à des erreurs de conception, mais à des erreurs de modélisation : il est loin d'être évident de donner une sémantique aux protocoles. Mieux encore, il n'existe pas à l'heure actuelle de langage de description spécifique aux propriétés sécuritaires et bien peu d'entre elles font l'unanimité quant à leur formalisation (en fait, même la confidentialité, qui recueille a priori le plus grand consensus est très différente d'un outil à l'autre si on y regarde de près).

Les outils disponibles traîtent du secret et/ou de l'authentification, comportent un analyseur de protocoles, qui fixe la sémantique utilisée et un démonstrateur qui, selon les cas, fera des hypothèses restrictives sur la classe de protocoles ou le type d'attaques, utilisera des approximations supérieures (s'il recherche une preuve) ou inférieure (s'il recherche une attaque).

Tous les outils automatiques font des hypothèses fortes sur les primitives cryptographiques :

L'hypothèse du chiffrement parfait qui suppose que le chiffré d'un message ne satisfait aucune autre équation que l'identité, autrement dit que deux messages chiffrés sont identiques si et seulement s'ils sont chiffrés avec la même clef et qu'ils chiffrent le même message

L'hypothèse du générateur aléatoire parfait qui suppose qu'il n'existe jamais de collision entre les nombres supposés engendrés aléatoirement

2.3 Les problèmes à résoudre

On peut classer ces problèmes en plusieurs catégories. Il y a des problèmes structurels de la recherche sur le sujet, que nous abordons dans le paragraphe 2.3.1, des problèmes d'harmonisation, que nous abordons dans le paragraphe 2.3.2 et enfin des problèmes de recherche. Nous proposons de découper ceux-ci en trois parties. D'abord le problème de la définition des propriétés de sécurité (paragraphe 2.3.3), puis la question de l'hypothèse du chiffrement parfait (paragraphe 2.3.4) et enfin la question de l'analyse de code de protocoles (paragraphe 2.3.5) .

2.3.1 Interactions des groupes de recherche et transfert

Il n'existe à l'heure actuelle aucune structure permettant de coordonner les recherches dans ce domaine et cela, ni en France, ni au niveau Européen, ni au niveau mondial (même si, au niveau mondial, il existe bien entendu des colloques spécialisés, qui sont aujourd'hui les seuls points d'échange).

L'absence de telles structures a plusieurs conséquences néfastes. Par exemple, le manque de visibilité, qui nuit au transfert technologique. Mais aussi, et on l'a vu ces dernières années, la multiplication des sémantiques et des outils ayant à peu près les mêmes objectifs.

Plusieurs actions seraient extrèmement utiles pour la coordination et la communication des équipes de recherche sur le sujet. Par exemple, la mise en commun des exemples de protocoles cryptographiques publiquement disponibles (le précédent catalogue, dû à Clark et Jacob, remonte à 1997). Autre exemple : répertorier les outils de vérification avec leurs caractéristiques exactes et les résultats obtenus. L'enjeu est, entre autres, la visibilité auprès des entreprises développant ou utilisant des protocoles sécuritaires.

2.3.2 Harmonisation des interprétations

Un des problèmes historiques de la vérification de protocoles cryptographiques est leur définition formelle. En effet, le catalogue de Clark et Jacob regorge de descriptions plutôt informelles qui ont toutes donné lieu à des attaques, souvent à cause précisément du manque de formalisme (mais aussi à cause d'erreurs de conception, bien sûr). C'est ainsi que le fameux protocole de Needham-Schroeder à clef publique à été prouvé correct avant qu'on ne trouve une attaque. Simplement parce que dans les deux cas, la sémantique n'était pas la même.

L'harmonisation de la syntaxe comme de la sémantique serait un grand progrès dans la coordination des efforts de recherche. Mais c'est malheureusement un objectif qui nous semble hors de portée, car il soulève de nombreuses querelles d'école. Sans chercher la standardisation, il semble néanmoins très utile d'identifier les points de divergence d'interprétation et d'avoir

un panorama plus clair des différentes options. A notre avis personne ne peut aujourd'hui se vanter de connaître et de comprendre les différentes possibilités de modélisation et les différents modèles proposés. Cette comparaison nous paraît en outre extrèmement souhaitable, non seulement du point de vue des questions traîtées, mais aussi du point de vue de l'efficacité des méthodes employées.

En outre, la mise en évidence des hypothèses effectuées permettra de mieux évaluer l'adéquation du modèle formel. Par exemple, et cela rejoint le point suivant, il n'est pas clair que certaines propriétés puissent s'énoncer simplement autrement que dans un modèle d'algèbre de processus.

2.3.3 Quelles propriétés de sécurité ?

Pire encore que les modèles de protocoles, la plus grande confusion règne en ce qui concerne les propriétés à vérifier. L'exemple de la vérification de propriétés temporelles devrait montrer la voie : après l'article historique d'A. Pnueli, on distingue bien aujourd'hui deux familles de logiques temporelles correspondant à deux modèles du temps. Mais concernant les propriétés sécuritaires, on est très loin de cette distinction claire : il n'existe aucun formalisme qui permette d'exprimer, même une fraction significative, des propriétés supposées être satisfaites par les protocoles. Par exemple, la propriété d'authentification (ou d'authenticité : le vocabulaire n'est même pas fixé) a fait couler beaucoup d'encre : au moins 8 versions différentes en ont été publiées. Et il ne s'agit que de l'une des propriétés les plus simples et les plus courantes. Bien peu se sont risqués par exemple à définir l'anonymat.

Un des problèmes majeurs est la multiplicité des formalismes de protocoles, qui sont aussi utilisés pour définir les propriétés. Une autre raison à cette cacophonie est la tentative d'utiliser les logiques épistémiques, qui s'est révélée (au moins dans la tentative BAN) finalement un échec.

Il s'agit donc d'un défi majeur : qu'est-ce qu'une propriété de sécurité ?

Il nous semble que la réflexion ne peut qu'être collective. L'objectif, au delà d'un analogue des logiques temporelles, est d'identifier les questions cruciales qu'il faut résoudre pour la preuve de telles propriétés. Par exemple, en logique temporelle, on sait que la vérification de modèles se ramène à un problème d'accessibilité dans un graphe (plus ou moins complexe selon les logiques). Quelles sont les propriétés qui peuvent se ramener à de l'accessibilité ? Comment ? Les réponses permettraient en outre de délimiter la portée de nos outils de preuve.

2.3.4 Affaiblir les hypothèses de chiffrement parfait

Une des critiques les plus fréquentes (et en outre adressée par les spécialistes de cryptologie) est la grossièreté des hypothèses effectuées, qui ne permettent pas de détecter des attaques répertoriées. On comprendra pourquoi ces hypothèses ont été prises, au vu des problèmes déjà évoqués dans les cas les plus simples. Les chercheurs en cryptologie possèdent des hypothèses plus réalistes, mais hors de portée d'une automatisation complète. Le défi est de savoir dans quelle mesure il est possible de s'en rapprocher dans la vérification automatique des protocoles.

Là encore, la confrontation de nos perspectives de recherche dans ce domaine serait très intéressante. A première vue, il y aurait deux directions possibles (que certains groupes ont commencé à regarder, mais il n'y a aucune publication sur le sujet à notre connaissance) : d'une part ajouter certaines propriétés algébriques pertinentes des primitives cryptographiques,

d'autre part utiliser des modèles probabilistes qui permettraient d'exprimer, par exemple, la très faible probabilité de collision des nombres engendrés aléatoirement.

2.3.5 Analyse de codes de protocoles

Finalement, comme l'ont montré les attaques récentes sur SSL, les faiblesses d'un protocole peuvent n'être ni au niveau des primitives cryptographiques, ni au niveau logique du protocole, mais bien dans sa réalisation. Détecter, par exemple, qu'un dépassement de capacité d'une zone tampon est incorrectement traîtée par une implantation logicielle est un problème qui, s'il est typique des protocoles cryptographiques, ne leur est pas spécifique.

Plus généralement, la vérification de protocoles cryptographique s'est concentrée sur des modèles très abstraits. Il faudrait, par des techniques d'abstraction (ou de concrétisation) pouvoir se rapprocher des programmes réellement utilisés.

Les problèmes à traîter rejoignent ici les objectifs de la vérification de cartes à puce et, plus généralement, de code mobile. Les techniques de typage et d'analyse statique sont bien sûr pertinentes et, ici, nous profiterions d'actions coordonnées avec les chercheurs dans les domaines connexes de la sécurité.

2.4 Ebauche d'un plan de travail

Notre objectif est de résoudre (autant que possible) les problèmes évoqués dans le paragraphe précédent. Quels sont les moyens que nous pouvons mettre en oeuvre pour favoriser la réalisation de cet objectif?

En dehors des réunions prévues de l'action, voici un certain nombre d'actions que nous nous proposons d'entreprendre ensemble :

1. Tout d'abord, très concrètement, et pour répondre à l'un des problèmes soulevés dans le paragraphe 2.3.1, nous nous proposons d'activer un nouveau catalogue de protocoles cryptographiques : nous mettrons en service dans les jours qui viennent une page web à Cachan qui, non seulement contiendra un répertoire mis à jour des protocoles, mais aussi de leurs preuves, attaques, commentaires. Les contributions (qui seront modérées) seront bienvenues de tous.
2. Nous nous proposons ensuite de mettre en service à Grenoble une page web qui sera le media de communication pour notre communauté. En particulier, elle donnera accès à nos travaux de synthèse (dont le premier est la présente ébauche), aux annonces de réunions pertinentes etc... Elle permettra de donner une visibilité à nos groupes de recherche.
3. Nous nous proposons de rédiger un état de l'art. Pour ne pas être trop ambitieux, il s'agirait plutôt de résumé, par chacune des équipes de recherche des fonctionalités des logiciels réalisés, des tests effectués et des programmes de travail. Ceci permettrait entre autres de favoriser la solution des problèmes soulevés dans le paragraphe 2.3.2.

3 Modèles pour la disponibilité et la survivabilité

3.1 Les enjeux

Les systèmes informatiques sont de plus en plus souvent la cible d'actions malveillantes, qu'il s'agisse de tentatives d'intrusion externes ou d'attaques internes. Néanmoins, ces systèmes doivent rester disponibles, c'est-à-dire continuer à fournir des services même s'ils sont attaqués. Il s'agit d'un enjeu important dont dépend en grande partie la crédibilité des futures infrastructures informatiques.

C'est dans ce contexte que le concept de survivabilité est apparu. Ce terme recouvre l'ensemble des mécanismes de sécurité qui permettent à un système informatique de résister à des attaques. Il s'agit cependant d'un concept flou. En effet, la plupart de ces mécanismes s'inspirent des principes proposés dans le contexte des fautes accidentelles et il n'est pas évident que de tels principes soient résistants à des actions intentionnellement malveillantes.

Il est donc nécessaire de définir précisément le concept de survivabilité. L'objectif de cet axe de l'action spécifique n'est pas de concevoir de nouveaux mécanismes visant à assurer la survivabilité. Il s'agit au contraire de modéliser et formaliser la propriété de disponibilité d'un système de manière à pouvoir vérifier que les mécanismes de survivabilité proposés par ailleurs permettent effectivement de garantir la disponibilité du système.

3.2 Le contexte

Avec la confidentialité et l'intégrité, la disponibilité est l'une des trois propriétés de sécurité de base. Cependant, alors qu'il existe plusieurs modèles pour les propriétés de confidentialité et d'intégrité, le concept de disponibilité n'a été que très peu étudié. On peut citer les travaux de modélisation réalisés par Yu et Gligor (1990) concernant l'expression de propriétés de disponibilité et ceux de Millen (1992) qui a proposé un modèle de disponibilité orienté allocation de ressources. Plus récemment, l'ONERA (1999) a travaillé sur l'expression de politiques et de propriétés de disponibilité et sur des techniques de model checking pour analyser la disponibilité d'un système.

Depuis quatre ou cinq ans, contrairement à la disponibilité, de très nombreux travaux ont été menés sur le thème de la survivabilité. En particulier, la DARPA finance plusieurs projets sur ce sujet. Les approches proposées dans ces projets s'inspirent en grande partie des techniques de sûreté de fonctionnement, telles que les mécanismes de tolérance aux fautes et la redondance passive ou active pour faire face aux attaques.

Plusieurs conférences ont déjà été organisées sur ce sujet, en particulier la DARPA Information Survivability Conference et l'Information Survivability Workshop (ISW) qui en est à sa quatrième édition (1997, 1998, 2000 et 2002).

En Europe, on peut citer les travaux menés dans le cadre du projet européen MAFTIA qui regroupe les universités de Newcastle, de Lisbonne et de Searland ainsi que le LAAS, QinetiQ et IBM Zurich. L'objectif de ce projet est de développer des approches tolérantes aux attaques reposant sur des mécanismes de confinement et de compensation des erreurs, sur la reprise du système en cas d'attaques et sur des méthodes d'injection de fautes dans le contexte de la détection d'intrusions. CSP est utilisé comme langage de spécification et les techniques de model checking sont proposées comme outil de vérification.

En France, il n'y a pas, à notre connaissance, de travaux traitant directement du problème de la survivabilité. On peut néanmoins citer les travaux menés dans le cadre du projet RNRT MP6 qui traite des problèmes de disponibilité dans le contexte des politiques de sécurité des

domaines de la santé et du social. Plus indirectement, on peut également citer le projet RNTL DICO qui s'intéresse, en particulier, à la détection des attaques en déni de service.

Rappelons cependant que l'objectif de cette Action Spécifique n'est pas d'étudier de nouvelles approches pour la survivabilité. Il s'agira plutôt de :

1. Approfondir les modélisations du concept de disponibilité
2. Recenser les mécanismes proposés pour assurer la survivabilité
3. Examiner les techniques permettant de vérifier qu'un mécanisme de survivabilité proposé permet effectivement de garantir certaines propriétés de disponibilité

3.3 Les problèmes à résoudre

La propriété de disponibilité n'ayant été jusqu'à présent que peu étudiée, il reste de nombreux problèmes à résoudre. Nous avons identifié quatre thèmes de recherche principaux :

1. Modélisation des propriétés de disponibilité
2. Concept de politique de disponibilité
3. Cohérence d'une politique de disponibilité
4. Techniques pour vérifier des propriétés de disponibilité

Nous allons, dans les sections suivantes, examiner brièvement ces quatre thèmes.

3.3.1 Modélisation des propriétés de disponibilité

Prévention du déni de service Dans les critères européens de la sécurité (ITSEC), la disponibilité est définie comme la protection contre les dénis de service. Un déni de service peut à son tour être défini comme une attaque ayant pour effet de réduire les performances du système. Selon cette première définition, la disponibilité peut donc être vue comme une *safety* property, c'est-à-dire une propriété qui garantit que quelque chose d'indésirable, le déni de service, ne pourra se produire dans le système.

Dans la pratique, on considère généralement que la plupart des protocoles utilisés sont vulnérables aux attaques en déni de service et qu'ils vont probablement subir de telles attaques. Il serait donc vain d'essayer de prouver que ces protocoles satisfont des propriétés de disponibilité. On essaye plutôt de détecter des attaques en déni de service contre ces protocoles. Il s'agit d'une part importante de la problématique traitée dans le cadre de la détection d'intrusion.

A notre connaissance, les travaux menés pour détecter des attaques en déni de service n'essayent pas de donner une modélisation formelle du concept de déni de service. Les approches classiques essayent plutôt de donner une description de l'attaque (la signature de l'attaque) qui va ensuite permettre aux outils de détection de reconnaître l'occurrence d'une telle attaque.

Cependant, les attaques en déni de service deviennent de plus en plus sophistiquées et donc complexes à détecter. En particulier, on voit apparaître la notion de déni de service « distribué » ; dans ce cas, le déni de service est obtenu par la combinaison de plusieurs attaques, chaque attaque prise séparément n'étant pas suffisante pour réaliser un déni de service.

Nous pensons que les approches classiques reposant sur l'expression de signatures ne sont plus adaptées pour détecter de telles attaques. La solution passe par la formalisation de la

notion de déni de service et par le développement de techniques permettant de déduire qu'une combinaison d'événements aura pour effet la réalisation d'une attaque en déni de service.

Garantie de la continuité de service Une autre approche consiste à définir la disponibilité comme la capacité du système à fournir des services aux utilisateurs autorisés dans des conditions d'utilisation données, en particulier, conditions d'horaires, de délai et de performances mais aussi conditions sur le système et l'utilisation des ressources. Dans ce cas, la disponibilité correspond à une *liveness* property, c'est-à-dire une propriété qui garantit que quelque chose de souhaitable, à savoir que le service sera rendu dans le respect des conditions, se produira dans le système.

Cette deuxième vision de la disponibilité vient renforcer l'approche par prévention du déni de service. En effet, un système qui serait vulnérable aux attaques en déni de service ne sera pas capable d'assurer ce service. En revanche, ce n'est pas parce que le système est protégé contre les attaques en déni de service, qu'il sera capable de fournir le service en respectant les conditions spécifiées.

3.3.2 Concept de politique de disponibilité

La garantie de la continuité de service repose sur la notion d'utilisateur autorisé et sur le fait que ces utilisateurs ont le droit d'attendre que le système leur fournissent certains services dans des conditions préalablement spécifiées. Ceci peut être représenté sous forme d'un ensemble de règles précisant, pour chaque service (ou classes de services) et chaque utilisateur (ou classe d'utilisateurs), les conditions que le système s'engage à respecter lorsque cet utilisateur demande la réalisation du service.

Cet ensemble de règles constitue une politique de disponibilité. Il s'agit d'une partie de la politique de sécurité au même titre que la politique de confidentialité et la politique d'intégrité.

La formalisation d'une politique de disponibilité combine des notions complexes : notions déontiques de droit, d'interdiction et d'obligation, notions temporelles et notion dynamique de service.

3.3.3 Cohérence d'une politique de disponibilité

Comme toute politique de sécurité, une politique de disponibilité doit être cohérente. Intuitivement, il s'agit de montrer que l'ensemble des règles constituant la politique de disponibilité ne contient pas d'exigence conflictuelle, c'est-à-dire qu'il serait impossible de satisfaire simultanément.

Des travaux ayant pour objectif d'analyser la cohérence d'une politique de sécurité ont déjà été réalisés. Compte tenu des particularités des règles constituant une politique de disponibilité, il s'agira d'examiner si ces travaux sont directement applicables dans ce contexte. Dans le cas contraire, on devra analyser les nouveaux problèmes que pose la disponibilité.

3.3.4 Techniques pour vérifier des propriétés de disponibilité

Etant donnée une spécification formelle d'un mécanisme de sécurité, l'objectif est ici de vérifier que cette spécification satisfait les exigences exprimées dans la politique de disponibilité.

A priori, ce problème n'a pas encore été examiné dans la littérature mais de nombreuses techniques de vérification sont candidates pour le traiter, en particulier celles utilisées dans le contexte de la sûreté de fonctionnement ou pour le test de performances.

L'objectif est donc d'expérimenter des techniques déjà existantes pour vérifier des mécanismes de survivabilité définis par ailleurs.

3.4 Ebauche d'un plan de travail

Notre objectif est d'analyser les problèmes évoqués dans la section précédente de manière à identifier les points durs et recenser les techniques candidates pour les résoudre. Les actions que nous comptons mener pour atteindre cet objectif sont les suivantes :

1. Effectuer un état de l'art. Sans être trop ambitieux, cet état de l'art cherchera tout d'abord à identifier les différentes définitions possibles des concepts de disponibilité et de déni de service, ainsi que les formalismes adaptés à leur expression. Nous chercherons également à recenser les principaux projets traitant du problème de survivabilité et présenterons, sous forme de résumé, les travaux menés dans ces projets. Enfin, nous identifierons les techniques pertinentes pour vérifier des propriétés de disponibilité ainsi que les équipes françaises susceptibles de maîtriser ces techniques.
2. Nous proposerons une étude de cas devant satisfaire des exigences de disponibilité élevées et examinerons, au travers de cette étude de cas, différentes techniques susceptibles d'être utilisées pour vérifier ces exigences.
3. Nous nous proposons enfin de mettre en service à Toulouse un site web qui hébergera les rapports de synthèse réalisés au cours de cette AS ainsi que les annonces de réunions pertinentes. Ce site pourra contribuer à donner une visibilité aux équipes de recherche travaillant en France sur les thèmes de la disponibilité et de la survivabilité.

4 Analyse de sécurité du code embarqué

4.1 Les enjeux

L'évolution récente vers des langages de programmation qui offrent des fonctions de sécurité propres suggère d'utiliser l'analyse statique pour la vérification formelle de propriétés de sécurité. Un exemple est le langage Java et son architecture de sécurité basée sur les règles de visibilité, l'attribution de permissions définies par l'utilisateur aux codes téléchargés et les contrôles dynamiques de sécurité par inspection de la pile.

La preuve de propriétés de sécurité de tels programmes nécessite de disposer de méthodes pour abstraire un programme en une version simplifiée qui peut ensuite être analysée. Pour ce faire il faut disposer de modèles abstraits qui permettent de se focaliser sur les aspects pertinents d'un programme. Plusieurs cadres ont été proposés, comme les graphes de flot de contrôle, le pi calcul ou encore les ambients mobiles. Un des objectifs de l'action sera d'évaluer et de comparer les avantages des différents cadres pour différents types de propriétés de sécurité.

4.2 Le contexte

Pour formaliser la notion de la sécurité informatique, un nombre important de *modèles de sécurité* ont été proposé. Parmi les premiers modèles de contrôle d'accès aux ressources d'un système on trouve le modèle de Bell et LaPadula composé de sujets et d'objets chacun avec sa classification dans un treillis de niveaux de sécurité (*public*, *confidentiel*, *secret*,...) et des règles interdisant certaines lectures et écritures selon les classifications des sujets et des objets. Depuis, les modèles ont évolués vers des modèles plus facilement liés avec un logiciel (on peut citer le modèle de non-interférence de Goguen et Meseguer et les travaux oxfordiens autour de CSP et la non-interférence).

4.2.1 Rapprochement avec les langages de programmation

Nous constatons tout de même qu'il est souvent difficile de trouver un modèle abstrait qui soit suffisamment riche pour pouvoir modéliser et raisonner sur un logiciel de taille non-triviale, écrit dans un langage de programmation de haut niveau. Une approche prometteuse a été initiée avec les travaux de Volpano et Smith autour des systèmes de type non-standard pour détecter des flux d'informations illicites. Comme dans d'autres travaux sur les types non-standard, l'idée de base consiste à enrichir les types standard d'un langage avec des propriétés décrivant le niveau de confidentialité d'une donnée. Une logique pour raisonner sur l'évolution des niveaux de confidentialité est traduit en un système de type pour le langage de programmation et les algorithmes d'inférence de type peuvent (non sans peine) être adaptés à l'inférence de ces types non-standard.

4.2.2 Sécurité et Java

Un défi supplémentaire en matière de sécurité logicielle vient des langages proposant leur propres dispositifs pour sécuriser un code. C'est le cas notamment avec Java qui proposent une architecture de sécurité complexe, fondée sur plusieurs notions comme les modificateurs de visibilité et la vérification dynamique des permissions d'un code. Nos propres travaux en la matière ont conduit à un modèle de programme basé sur la notion d'automates à

pile. La modélisation d'un code Java se fait à travers une analyse statique qui extrait une approximations finitaire du flot de contrôle du code. Le résultat peut ensuite être soumis à la vérification (par *model checking*) de propriétés sécuritaire exprimées dans des formalismes comme les automates finis ou la logique temporelle. Un problème important à étudier dans le cadre de l'Action Spécifique est comment rapprocher ces modèles à d'autres types de politiques de sécurité qui ne sont pas forcement exprimées dans un des formalismes mentionnés plus haut.

4.2.3 Critères communs

L'intérêt de disposer de méthodes de vérification avec des bases formelles a été renforcé avec l'arrivée de normes industrielles pour évaluer la sécurité d'un système informatique. Les *Critères Communs* permet d'attribuer un niveau de sécurité, où un niveau plus élevé se traduit en des exigences plus sévères en terme d'utilisation de méthodes formelles dans le processus de génie logiciel. Les validations par les Critères Communs se font principalement sur les logiciels critiques où une défaillance peut avoir des conséquences graves. Un secteur qui est fortement demandeur de ce type d'évaluation est l'industrie de cartes à puce. On note, cependant, que le choix de modèles et de techniques de preuve à utiliser dans ces évaluations est lion d'être fait. L'Action a donc une occasion pour influencer ces choix en proposant des méthodes qui à la fois reposent sur des bases formelles ioconstestables et repondent aux besoins concrèts des industriels. A ce fin, nous comptons profiter de nos contacts industriels déjà établis pour cibler nos efforts.

4.3 Logiciel embarqué sur une carte à puce

Pour à la fois profiter des expertises acquises et mieux focaliser nos activités, nous allons nous concentrer sur le logiciel pour un type de système embarqué particulier : la carte à puce. Une des motivations pour ce choix est l'existence d'un langage de programmation standardisé pour ce type d'applications. Le langage Java Card a été proposé comme un langage de haut niveau qui permet un développement plus simple et plus sûr d'applications pour les cartes à puce. Il s'agit d'une version réduite de Java dotée des extensions propres aux cartes à puce. Plusieurs types de données (comme les chaînes de caractères et les nombres flottants) jugés trop gourmands en mémoire ont été supprimé. La notion de *threads* et la récupération de mémoire automatique ne figurent pas non plus dans le langage. En revanche, des méthodes pour gérer des objets persistants et des facilités pour définir des "transactions" ont été ajoutées.

Grâce à l'architecture de la machine virtuelle Java Card il est possible de gérer plusieurs applications sur une même carte. il en va de soi que cette facilité pose des soucis en terme de sécurité des applettes coopérant sur une carte. Pour répondre à ces soucis les concepteurs ont repris certaines parties de l'architecture de sécurité du langage Java comme le typage fort (pas de manipulations de références) et les modificateurs de visibilité qui permettent de limiter les accès aux membres d'un objet.

Le gestionnaire de sécurité étant trop encombrant pour être implanté sur une carte à puce, les concepteurs de Java Card ont ajouté un pare feu entre les applettes. Ce pare feu interdit *a priori* tout accès aux membres d'applettes appartenant à un contexte autre que le contexte de l'applette qui tente l'accès. Ce dispositif, trop contraignant pour permettre une bonne coopération entre applettes, a ensuite été enrichi avec avec la possibilité de déclarer des *objets partagés* auxquels toute applette puisse accéder.

Devant cette multitude de dispositifs sécuritaire, il est naturel de se poser la question : comment s'assurer que les dispositifs employés dans les applettes installées sur une carte fournissent un niveau de sécurité (en terme de confidentialité, d'intégrité et de disponibilité) requis. Notre réponse à cette question et l'objectif global des activités sera de fournir des méthodes de validation de la sûreté et la sécurité d'applications Java Card.

4.4 Ébauche d'un plan de travail

- Établir une formalisation commune des aspects de sécurité propre au langage Java Card. La séparation des applettes par des pare-feu ainsi que les mécanismes de communication entre applettes dans la machine virtuelle de Java Card nécessitent des techniques d'analyse de flot de données et de contrôle pour obtenir une modélisation précise du pare-feu Java Card.
- Formaliser des politiques de sécurité pour une carte à puce. Cette partie s'intègre naturellement dans activité autour des politiques de sécurité. L'objectif ici est d'exprimer les politique dans un formalisme qui ensuite permet de vérifier qu'un ensemble d'applettes respectent une politique donnée. Nous étudierons notamment comment les politiques de disponibilité (voir la partie *Modèles pour la disponibilité et la survivabilité*) peuvent être liées aux modelés abstraits des applications Java Card.
- Étudier comment étendre et adapter des méthode de vérification générique aux spécificités de Java Card. Nous pensons en particulier nous appuyer sur les techniques de model-checking et la preuve de programmes pour prouver des propriétés des applettes Java Card. Nous nous intéressons aux propriétés de sûreté et de bonne formation du code (pas de transactions imbriquées, pas d'exceptions non attrapées, *etc*) et des propriétés comme le flux de données et d'objets sur la carte.
- Une attention particulière sera apporté au problème de flux d'information entre applette. Plus précisément, nous nous proposons de définir une analyse de confidentialité pour des applettes Java Card, dont les données sont classifiées dans un treillis de niveau de sécurité. L'analyse permettra d'établir l'absence de flux d'information illicite entre les applettes (on parle alors de propriété de non-interférence), et sera exprimée comme une extension du système de types pour Java Card avec des annotations de niveau de sécurité. Dans un premier temps, nous envisageons de définir une analyse simple (monovariante) et de l'appliquer à des scénarios multi-applications. Sur la base des résultats obtenus, nous étendrons notre approche à des analyses plus précises mais plus complexes. A terme, nous souhaitons formaliser cette analyse dans un assistant à la preuve, et établir sa correction

4.5 Diffusion de connaissances

Nous allons profiter de cette action pour fédérer et consolidier les connaissances autour de Java Card. Cette fédération se fera à travers la création d'un site web qui

- recense les différents sites relatifs à la technologie Java Card (universitaires, industriels, personnels, ...)
- regroupe les différents travaux de recherche sur la technologie Java Card avec si possible le maximum d'article
- détaille les différents outils existants pour faire de la vérification de propriétés sécuritaires sur la technologie Java Card.

– met à disposition les rapports issus de l'action au fur et à mesure que ceux-ci atteignent un niveau de maturité qui permet leur publication.

Ce site sera (pour une grande partie) ouvert à accès depuis l'extérieur.