REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MENTOURI CONSTANTINE
FACULTE DES SCIENCES DE L'INGENIEUR
DEPARTEMENT D'INFORMATIQUE

Thèse de

DOCTORAT EN SCIENCES-INFORMATIQUE

Thème :

Résolution du Problème des Hôtes Malicieux : Une Approche Uniforme Basée Médiation

Présentée par : Mr. Chihoub Mohammed

Devant le jury

Président : Pr. Boufaida Mahmoud. Professeur, Université Mentouri - Constantine.

Rapporteur : Dr. Maamri Ramdane. M.C, Université Mentouri - Constantine.

Examinateurs: Pr. Bilami Azzeddine. Professeur, Université de Batna.
Pr. Benmohammed Mohamed. Professeur, Université Mentouri - Constantine.
Dr.Mokhati Farid. M.C, Université de Oum-El-Bouaghi.

Soutenue En Date du: 23/10/2012

Dédicaces

A la mémoire de mon défunt père, mort pour la patrie. Cette thèse est sans conteste un fruit de son sacrifice. Qu'il le savoure fièrement au paradis !

A ma mère, toujours dévouée et infiniment altruiste, qui, grâce à son amour et son éducation parfaite, a su nous guider, mes frères, mes sœurs et moi, durant toute notre vie. Qu'elle trouve dans l'accomplissement de cette thèse une expression de ma reconnaissance éternelle,

A mon épouse, qui m'a agréablement accompagné durant ma recherche et a partagé tous les palabres de cette thèse, A mes merveilleux enfants, Dhoha, Chiheb et Abdou,

A mon frère aîné, mais aussi mon Maître, Messaoud, et sa petite famille,
A mon frère Abdelaziz et sa petite famille,
A mes sœurs et leurs petites familles,
A toute ma famille,
A tous mes amis.

Remerciements

La réalisation de mon doctorat est une tranche de ma vie à part entière. Beaucoup de personnes ont participé directement ou indirectement à sa concrétisation et méritent d'être remerciés quel qu'en soit leur degré d'investissement.

D'un point de vue professionnel, je commencerai par remercier mes directeurs de thèse successifs : Dr. Maouche Mourad, Pr. Sahnoun Zaidi, Dr. Maamri Ramdane pour le travail qu'ils ont effectué à mes côtés et qui était emprunt tout autant de conseils pour orienter mes recherches que de critiques nécessaires à la pertinence de mon étude.

Avoir l'appui d'un tel panel ainsi que celui du Pr. Boufaida Mahmoud qui m'a fait l'honneur pour avoir accepté de présider mon jury est certainement un atout.

Ensuite, je souhaite remercier les membres de jury en l'occurrence le Pr. Bilami Azzeddine, le Pr. Benmohamed Mohamed, et le Dr. Mokhati Farid d'avoir accepté d'évaluer mon travail et pour leurs éventuels commentaires.

Que le Professeur, Kamel Barkaoui, du Cnam de Paris, trouve ici le témoignage de ma profonde gratitude et mes vifs remerciements, lui qui m'a accueilli à cœur et bras ouverts, depuis de longues années déjà dans son Laboratoire, malgré mes quelques maladresses d'occasion.

Je tiens aussi à exprimer ma gratitude envers : Dr. Belala Faiza, Chellali Nadia, et Habes Abdellatif qui ont eu la gentillesse de lire de nombreux morceaux si ce n'est l'intégralité de mon manuscrit et qui ont usé de leur expertise pour déceler certains aspects qui m'avaient échappés.

Je tiens aussi à remercier de nombreux collègues qui ont partagé mon quotidien. En particulier, Sebih Hacene, Laraba Mohamed El-Habib, Derdouri Lakhdar, Zarour NacerEddine, Bouznada Mourad, Bouznada Nacer , Saidouni Djamel, Mezioud Chaker, Zitouni Abdelhafid et tant d'autres. Malheureusement, ici tout le monde ne peut apparaître nommément et je me dois de faire une sélection quoique subjective.

Je n'oublierai pas les collègues de l'équipe de recherche GLSD du Laboratoire Lire de l'Université Mentouri de Constantine avec qui j'ai partagé de bien agréables séminaires.

Je n'oublierai pas non plus tous les collègues du Département et sans exception pour le soutient et les encouragements qui m'ont toujours apportés. A vous tous merci, sans ces moments de connivence et votre confiance en moi, je n'aurais sans doute jamais mené à bien cet ouvrage.

Abstract:

When mobile agents emerged, they had exalted many hopes among researchers. Expectations from mobile agents were huge. Mobile agents' paradigm is the plausible candidate model to substitute the classical model of client/server. The idea is to replace client data fetching by moving an agent to the data remote host. The so called mobile agent is a software able to move around during execution. The paradigm of mobile agents has many advantages such as asynchronous execution, autonomy, reducing network load, and also the ease to update agents themselves. Mobile code is considered by many researchers as the approach for better: (1) Configurability. (2) Scalability. (3) Customizability. However, mobile agents are facing problems for widespread use and deployment. Developing with mobile agents requires a secure architecture underneath; especially for e-commerce applications. Two major problems are to consider when dealing with mobile agents security: (1) Platforms protection, and (2) Mobile agents' protection. Protecting mobile agents from malicious hosts' attacks is by far the most difficult problem to resolve. The subtle problem of malicious hosts in Internet makes the use and deployment of mobile agents a hazardous operation. Our main objective is to propose a process to clean up Internet from such kind of hosts. In fact, the chaotic situation made by the presence of malicious hosts is unacceptable. This thesis proposes a uniform approach against tampering host phenomena struggling. The approach uses a mediation entity in two solution types. One of the objectives of this thesis is to show the benefit of the standardization of malicious hosts' problem solving using a mediator. At this level, the mediation is integrated both in detection and preventive techniques. The second objective purpose is to show how mediation contributes simplifying complexity of protocols based on detection. The impact of mediation on such protocols complexity is made via the development of brand new protocol. Mediation integrated to Minsky preventive protocol is able to: (1) Determine malicious hosts and revoke probably which are recidivist. (2) Disclose bizarre behavior of trusted hosts which can be disseminated in the net. The basis of this mediator is the formal frame work of oo-action systems. Unlike, in the proposed and validated brand new protocol, the mediator has more powerful capabilities. They are close to the Truth extraction frontier. In its preliminary stage, the protocol collects easily a series of hints. These hints enable discrimination between safe and suspect executions. The mediator explores the confidence of the claims and deduces tampering or tampering free acts of hosts when finally Truth establishment process is activated.

Key-Words: Mobile Agents, Secure Architecture, Malicious Hosts, Mediation, oo-action Systems.

Résumé :

Dès son émergence, le paradigme agent mobile a nourri beaucoup d'espoirs chez les chercheurs. Les attentes des agents mobiles sont grandes et multiples. Les agents mobiles forment un modèle candidat plausible pour se substituer au modèle client/serveur. L'idée est de remplacer l'envoi de données vers le client par le déplacement d'un agent vers l'hôte possédant les données pertinentes. Un tel agent dit mobile est un software capable de migrer d'un ordinateur à un autre lors de son exécution. Ce paradigme présente beaucoup d'avantages tels que l'exécution asynchrone, l'autonomie, la réduction de la charge du réseau, et la facilité de mise à jour de l'agent lui-même. La mobilité du code est considérée par beaucoup de chercheurs comme l'approche la plus prometteuse aux problèmes : (1) de souplesse de configuration ''Configurability''. (2) de souplesse de croissance ''Scalability''. (3) de personnalisation ''Customizability''. Cependant, les agents mobiles n'ont pas eu l'essor qui leur a été prédestiné, d'emploi et de déploiement massifs. Le développement à base d'agents mobiles requiert une architecture sous-jacente sécuritaire; particulièrement dans le cas des applications du e-commerce. Deux problèmes majeurs sont liés à l'aspect sécuritaire des agents mobiles : (1) La protection des hôtes d'attaques des agents. (2) La protection des agents mobiles d'attaques des hôtes dits malicieux. Le problème de la protection des agents mobiles contre les attaques d'hôtes malicieux est beaucoup plus difficile à résoudre. Le problème délicat des hôtes malicieux dans Internet rend l'utilisation et le déploiement des agents mobiles une opération hasardeuse. L'objectif principal de notre thèse est de proposer un procédé d'assainir Internet de ces Hôtes. La situation chaotique que provoquerait la présence de tels hôtes est sans dire inadmissible. La thèse propose une approche uniforme de lutte contre le phénomène des hôtes malveillants. L'approche préconise l'utilisation d'une entité de médiation dans deux solutions. L'un des objectifs de la thèse est de montrer la latitude de justifier la standardisation de la résolution du problème des hôtes malicieux par la médiation. A cet effet, la médiation est intégrée dans deux protocoles appartenant aux deux familles de techniques préventives et de détections. Le deuxième est de montrer que la médiation contribue aussi à simplifier la complexité inhérente aux techniques basées sur la détection. L'influence de la médiation sur la complexité de tels protocoles est exemplifiée via la construction et la validation d'un nouveau protocole. Le médiateur de l'extension de la technique préventive de Minsky, proposé dans cette thèse, est capable de : (1) déterminer les hôtes malicieux et d'en révoquer probablement ceux qui sont récidivistes. (2) de discerner les comportements bizarres des hôtes de confiance et de faire en sorte qu'ils soient disséminés. L'assise de ce médiateur est le cadre formel des systèmes oo-action. Par contre, dans le nouveau protocole construit et validé, des compétences plus sophistiquées sont octroyées au médiateur. Ces capacités approchent la limite du niveau de l'extraction de la vérité. Dans sa phase préliminaire, le protocole collectionne de manière simple et facile une série d'indications. Ces indications sont utiles pour discriminer entre les exécutions qui sont intègres de celles qui sont suspectes. L'investigation par le médiateur explore le degré de confiance des acclamations et d'en déduire en fin l'occurrence ou non d'actes malicieux des hôtes par l'amorçage d'un procédé d'établissement de la vérité.

Mots-Clés : Agent Mobile, Architecture Sécuritaire, Hôtes Malicieux, Médiation, oo-action Systèmes.

ملخص:

ساد لدى الباحثين اعتقاد حال ظهور العملاء الجوالين ,أنهم الحل الأمثل لعدة مشاكل عويصة . فعلا إن ما هو منتظر من العملاء الجوالين لهو كبير. نموذج العملاء الجوالين للبرمجة هو بالتأكيد مرشح لأخذ مكان نظام البرمجة الكلاسيكي المعتمد ألا و هو زبون / خادم . الفكرة الأساسية هنا, هي القيام بتعويض إنزال المعلومة عند الزبون بتحويل العميل الجوال إلى مكان المعلومة عند الخادم. إن نموذج العملاء الجوالين له عدة مزايا من بينها: المعالجة غير المتزامنة , الاستقلالية, خفض حجم الحمولة بالشبكة وأخيرا سهولة تحيين العميل الجوال بحد ذاته. بمنظور عدة باحثين فإن الكود الجوال هو مقاربة لتحسين: (1) التشكيل Configurability.(2) مرونة التوسع Scalability .(3) التفصيل Customizability . إلا أن العملاء الجوالين يحتاجون إلى هندسة قاعدية آمنة؛ بالأخص في التطبيقات الخاصة بالتجارة الإلكترونية. مشكلان هامان ؛ يجب بحثهما عند التطرق لأمن العملاء الجوالين : (1) حماية منشآت تنفيذ العملاء الجوالين (2) حماية العميل الجوال نفسه . إن حماية العملاء الجوالين من هجومات المنشآت ذات المنحى الخبيث هو بالأخص أصعب مشكلة أمنية تتطلب الحل. إن وجود هذا المشكل على نطاق الانترنت يجعل استعمال و انتشار العملاء الجوالين عملية محفوفة بالمخاطر. هدفنا الأساسي في هذه الأطروحة ،هو اقتراح سيرورة عمل لتنظيف الانترنت من هذا النوع من المنشآت .فعلا الوضعية الحرجة الناتجة عن وجود تلك المنشآت هي غير مقبولة على الإطلاق . لهذا السبب تقترح الأطروحة مقاربة موحدة لمحاربة ظاهرة نشاط هذا النوع من الأعمال الضارة, حيث أن المقاربة المقترحة تستعمل آلية الوساطة لإيجاد حل لهذا المشكل. إن أول هدف لأطروحتنا هو طرح الوساطة كأداة معيارية لإنتاج الحلول المرجوة .في هذه الحالة الوساطة تستعمل كآلية موحدة في نوعين من الحلول : الوقائية، و العلاجية. الهدف الثاني يهتم بأثر آلية الوساطة على تخليص الحلول العلاجية من تعقيداتها. لتبيان ذلك تم تطوير بروتوكول جديد ليبرهن على ذلك. إن آلية الوساطة الملحقة ببروتوكول Minsky الوقائي تمكن من : (1) إيجاد المنشآت ذات المنحى الخبيث و إقالتها عند الضرورة و ذلك عند تكرار الأمر . (2) الكشف عن الأعمال غير المنضبطة, الخاصة بالمنشآت ذات الثقة و نشرها على الشبكة .إن الأساس لآلية الوساطة هاته هو أساس رياضي مبني داخل الإطار الرياضي العام لأنظمة oo-action. على الخلاف, في البروتوكول المعالجتي الجديد، و المقترح في هذه الأطروحة، فإن قدرات آلية الوساطة قد طورت حيث أنها تقترب من حدود استخلاص الحقيقة .في مرحلته الأولى البروتوكول يقوم و بسهولة واضحة بجمع سلسة من المؤشرات .هذه المؤشرات تمكن من التفريق بين المعالجات الآمنة للعميل الجوال من التي هي مشكوك في أمرها. إن الوسيط المعتمد في هذه الحالة يقوم بالتأكد أولا من درجة الثقة الموجودة بالشكاوي ,وأخيرا يتأكد من وجوده أو عدمه للأعمال ذات المنحى الخبيث و ذلك لحظة تنشيط سيرورة استخلاص الحقيقة .

الكلمات المفتاحية *: العملاء الجوالون , هندسة آمنة , منشآت ذات منحى خبيث, آلية الوساطة , أنظمة oo- action.*

Table des Matières

Chapitre 3 Protocole Résilient aux Erreurs Etendu

Chapitre 4 Traces Cryptographiques et Limitation du Temps d'Exécution

Chapitre 5 Nouveau Protocole de Détection des Hôtes Malicieux

Liste de Tables & Figures

Introduction Générale

Les systèmes ouverts tels que l'Internet exigent un effort et un soin particulier dans la conception et la programmation d'applications réparties où l'accès aux ressources et à l'information doit être permanent. En effet, la plus part de ces applications sont basées sur l'architecture client/serveur traditionnelle dont le caractère statique se prête mal à l'aspect dynamique et à la diversité de tels systèmes.

L'espoir né de l'apparition des agents mobiles en 1994 s'est vite confronté aux problèmes sérieux de sécurité. Les attentes engendrées par les agents mobiles sont grandes et multiples. Les agents mobiles forment un modèle candidat pour substituer le modèle client/serveur. L'idée est de remplacer l'envoi de données vers le client par le déplacement d'un agent vers l'hôte possédant les données pertinentes. Un tel agent dit mobile est un software capable de migrer d'un ordinateur à un autre lors de son exécution [1]. Ce paradigme présente beaucoup d'avantages tels que l'exécution asynchrone, l'autonomie, la réduction de la charge du réseau, et la facilité de mise à jour de l'agent lui-même. Ceci a motivé le développement d'applications à base d'agents mobiles dont les plus prometteuses sont celles relatives au e-commerce [2]. D'autres applications, susceptibles d'avantager l'utilisation des agents mobiles, sont celles en relation au sea-of-data, où une masse de données distribuée sur plusieurs serveurs ne peut être transférée via le réseau et ce pour diverses raisons telles que les contraintes liées à la bande passante ainsi que les restrictions légales [3]. En effet, le traitement local des données par l'agent mobile est axé sur les techniques du data mining ou sur des algorithmes d'apprentissage des machines.

Le paradigme agent mobile s'adapte bien à la nature non stable et transitoire des canaux de communication et fournit des performances meilleures particulièrement dans des réseaux non fiables à capacité de transfert limitée. Des études ont montré que la technologie des agents mobiles permet une abstraction intuitive, simplifiant ainsi la conception de plusieurs applications distribuées [4]. De ce fait, les agents mobiles devront jouer, selon l'avis de plusieurs chercheurs, un rôle important dans l'avenir du traitement distribué. Cependant, malgré ses nombreux avantages, le paradigme des agents mobiles n'a pas encore connu l'essor escompté. Le manque d'intérêt porté à ce paradigme est dû aux aspects sécuritaires qui posent des problèmes subtils. Ainsi, le développement d'applications distribuées est effectué à

base d'autres paradigmes où il y a moins d'avantages et de défis [5]. Le développement d'applications distribuées à base d'agents mobiles requiert une architecture sous-jacente sécuritaire; particulièrement dans le cas des applications du e-commerce.

Plusieurs avancées dans le domaine de la sécurité, la fiabilité et l'efficacité de la technologie agents mobiles ont été réalisées [6, 7, 8]. Il n'en demeure pas moins que, l'aspect sécuritaire des agents mobiles représente un frein considérable à son adoption à grande échelle [10]. Deux problèmes majeurs liés à l'aspect sécuritaire des agents mobiles sont : la protection des hôtes d'attaques des agents et la protection des agents mobiles d'attaques des hôtes dits malicieux.

Protéger les hôtes est un sujet dont la solution repose en partie sur les techniques de sécurité des systèmes conventionnels tels que les moniteurs de références. Plus récemment, d'autres chercheurs ont proposé de nouvelles techniques telles que le code avec preuve ou le ''sandboxing'' .i.e. carré de sable basée sur une limitation des accès de l'agent.

Le problème de la protection des agents mobiles contre les attaques d'hôtes malicieux est beaucoup plus difficile à résoudre [9]. Plusieurs solutions ont été proposées sans qu'aucune ne soit pleinement satisfaisante. Ces solutions varient considérablement, d'un niveau purement matériel jusqu'au niveau purement logiciel. Sans échapper à la tradition, les deux grandes orientations des solutions sont : la prévention et la détection.

L'existence et la prolifération des hôtes malicieux rajoutent à la complexité de gouvernance d'Internet, une nouvelle dimension. Le problème des hôtes malicieux est un obstacle majeur à l'adoption du paradigme des agents mobiles. Dans un système ouvert comme Internet qui de par sa nature est agressif ; la recherche d'une solution au problème des hôtes malicieux (i.e., la protection des agents mobiles) a suscité beaucoup d'intérêt parmi les chercheurs. Assainir un tel système ouvert des hôtes malicieux est indispensable pour éviter une situation chaotique. Les solutions dont découle ce processus font partie de deux grandes familles de protocoles qui utilisent soit la prévention ou la détection. Malgré que, ce processus peut se manifester par un effet de bord dans les protocoles de prévention, il est la consécration des protocoles de détection. Dans ces derniers, la vérification de la véracité de la preuve du caractère malicieux d'un hôte est toujours entreprise par une tierce partie de confiance. De

notre avis, l'intérêt de ce processus est de réduire la complexité des solutions de protection des agents mobiles.

Il est admis par plusieurs chercheurs que les techniques fournissant une protection entière de l'agent mobile sont en pratique difficiles à réaliser [5]. Pour cela, les efforts doivent se concentrer sur des solutions partielles, qui préviendront un sous-ensemble d'attaques, plutôt que de protéger complètement l'agent mobile. Nous pensons qu'une solution simple et efficace de protection entière ne soit pas impossible. Nous, nous intéressons particulièrement aux techniques qui améliorent la survivance d'un propriétaire des agents mobiles. Ces techniques se regroupent dans l'approche de détection d'actes malicieux survenus [11]. Dans les solutions que nous proposons, un rôle majeur est réservé à la tierce partie de confiance (Third Trusted Party i.e., TTP), entité capable de résoudre et de décider de la pertinence des requêtes de conflits potentiels. Nous avons prévu de donner de façon précoce comme rôle à la TTP celui d'un médiateur dont les compétences sont plus étendues. Les preuves des actes malicieux des solutions de détection actuelles ne sont fiables qu'à un coût très élevé [12,13]. Ce coût est dû au fait que la tierce partie de confiance ne fait que proclamer la sanction dans le cas où le caractère malicieux venait d'être confirmé par les faits disposés dans la preuve. Cette faculté simple de proclamation de la sanction uniquement, ne fait que maintenir la complexité des solutions à proposer dans le futur. Le risque de faire des extra vérifications systématiques inutiles par manque d'indications, ne fait qu'augmenter ce coût. Le cas des traces cryptographiques proposées par Vigna[14] en est un exemple.

Epurer Internet des hôtes malicieux requiert des pré-arrangements et une orchestration de la part de la tierce partie de confiance qui jouera un rôle de médiateur [44, 12]. L'objectif de cet assainissement est de simplifier les protocoles de la détection des actes malicieux dont la fiabilité est affaiblie. Ces protocoles doivent écarter le recours systématique à la vérification. Un constat préalable à base d'indications faciles à collecter pendant le séjour de l'agent mobile est accompli. Le but est de discriminer les exécutions intègres et d'en extraire que celles où le doute subsiste. Le procédé complexe et plus élaboré de médiation est apte ensuite à confirmer ou à infirmer les actes malicieux en vérification.

Dans une première partie de notre contribution, nous avons sélectionné un protocole de prévention auquel nous intégrons le procédé de médiation. Celle-ci est déclenchée comme un processus d'effet de bord. Le protocole utilise le vote et la duplication pour garantir la survie

de l'agent mobile vis à vis aux comportements malicieux des hôtes visités. Bien qu'intuitivement le protocole apparaît lourd, les expériences menées par les auteurs [15, 16] indiquent que la duplication améliore les performances des applications. Notre choix est guidé par la simplicité de la mise en œuvre d'un tel protocole en une infrastructure des plus courantes avec un potentiel réel quant à la traçabilité d'actes malicieux. Les autres protocoles, en particulier '' tamper free hardware '' ou '' mobile cryptography '', sont écartés car : (1) l'expérimentation est complexe et nécessite un hardware spécial non disponible jusqu'aujourd'hui, vis-à-vis du premier protocole. (2) le code se limite uniquement aux fonctions polynomiales et rationnelles en ce qui concerne le deuxième. (3) et plus important encore, les deux protocoles partagent un manque flagrant de traçabilité dû au confinement des exécutions de l'agent mobile. Nous avons intégré au protocole [15, 16] un procédé de médiation permettant l'assainissement des hôtes malicieux. Ce procédé est développé sur des bases mathématiques dans un cadre formel qui est le '' oo-action ''. L'exactitude certaine des traitements des applications protégées par un protocole préventif, simplifie le rôle du médiateur. En effet, il se caractérise uniquement par une simple dissuasion via des avertissements propulsés par un processus de gestion de réputation des hôtes.

La deuxième partie de notre contribution est consacrée aux protocoles de détection des hôtes malicieux. Ces derniers sont plus viables que les protocoles de prévention [11]. Une tierce partie de confiance se charge de résoudre les conflits potentiels. Les protocoles actuels ne fournissent de preuves fiables qu'à un coût très élevé. Nous avons opté pour une démarche qui réduira la complexité de tels protocoles. Cette réduction dépend de la capacité d'injecter dans la tierce partie de confiance des compétences plus sophistiquées. L'exploit réel de ce principe se concrétise pleinement dans notre protocole caractérisé par sa robustesse et sa manière incrémentielle d'agir [13]. Dans ce protocole la tâche de la tierce partie de confiance est plus élaborée. La sophistication approche la limite d'un exploit réel où les capacités de la tierce partie de confiance atteignent le niveau de l'extraction de la vérité. Le protocole proposé exploite dans sa première phase de collecte d'indications, un principe simple et efficace : ''toute action entreprise nécessite un temps petit aussi infime que soit-il''. Les actes malicieux ne dérogent pas à la règle et donc, ces actes qui sont des actions extra au travail typique accompli par l'agent mobile dans les sites de son itinéraire, sont facilement détectés à condition qu'ils ne soient pas fugitifs. On entend par '' fugitif '' le fait qu'ils s'inscrivent dans la tâche préconisée par l'agent et n'altèrent en rien la cohérence de son état, tel un envoi à une

destination non programmée. Le protocole développé découle en partie des deux protocoles qui seront présentés plus tard, à savoir les traces cryptographiques de Vigna et la limitation du temps d'exécution d'Esparza [14, 17].

La suite de cette thèse est organisée en cinq chapitres. Le chapitre 1 expose la problématique de la sécurité dans le paradigme agent mobile et les principales méthodes de résolution proposées dans la littérature. Le chapitre 2 etudie le protocole de Minsky et le cadre formel ''oo-action''. Le chapitre suivant décrit l'intégration de la médiation au protocole de Minsky ainsi qu'une description des attaques et le comportement du protocole amélioré dans chacun des cas. Le chapitre 4 est consacré aux travaux connexes particulièrement ceux de Vigna et Esparza qui nous ont conduit à proposer un nouveau protocole robuste et incrémentiel de détection des hôtes malicieux. Le chapitre 5 est consacré à l'étude détaillée du protocole conçu. Cette étude intègre les principaux concepts de base et la démarche de résolution du problème d'une protection entière de l'agent mobile. Le chapitre 5 développe aussi l'étude expérimentale renforçant notre hypothèse. Elle se compose d'un choix d'un exemple représentatif, de l'infrastructure de l'environnement de développement, et de l'approche statistique de validation. Une attention particulière est réservée à la tâche très élaborée de la TTP. Dans la conclusion générale nous résumons les principaux résultats obtenus, nous faisons état de la partie du protocole absente dans l'implémentation avec des indications sur sa réalisation et nous présentons enfin des suggestions et des orientations pour des travaux futurs.

Chapitre 1
Agents Mobiles et Sécurité

1. Introduction

Dès sa naissance, l'informatique n'a cessé d'évoluer et les systèmes se sont peu à peu organisés en réseau. L'émergence des systèmes ouverts dans les entreprises ou à l'échelle d'Internet a donné lieu à de nouvelles orientations dans la recherche. Dans ce contexte les soucis de passage à l'échelle, la personnalisation de services, la flexibilité, l'autonomie, la tolérance aux pannes, le support de nœuds sans fils, et le support des opérations en mode déconnecté sont des facteurs dominants. Conceptuellement, la mobilité du code constitue une réponse appropriée à ces préoccupations. En effet, la migration du code dénommé ''agents mobiles'' est un pas en avant envers ces aspirations et elle a donné lieu à de nombreuses recherches aussi bien dans les milieux universitaires que corporatistes. Le résultat est que plusieurs applications basées agents mobiles ont vu le jour dans des domaines tels l'administration de systèmes, les télécommunications, et le e-commerce où l'agent cherche le meilleur prix pour un produit [18, 19**].** Cependant, pour que ces applications deviennent des tâches courantes, elles doivent se déployer sur des plates-formes d'agents mobiles sécuritaires. Dans ce chapitre, avant d'analyser la question de la sécurité des agents mobiles, nous abordons de manière brève et succincte le concept de l'agent et l'aspect de la migration du code. Nous examinons par la suite les techniques de protection de la plate-forme puis celles de protection de l'agent, enfin, nous terminons par une synthèse des problèmes ouverts.

2. Qu'est ce qu'un agent ?

Le terme ''agent'' a une signification qui est vague. Ses diverses définitions découlent, selon l'avis de Stan Franklin et al[46], directement de l'ensemble des exemples imaginés par un chercheur. A travers une définition particulière, nous traduisons une instanciation de ces exemples.

Chez Wooldridge et Jennings [47], un agent est un matériel et plus particulièrement un logiciel avec les propriétés suivantes:

1. L'autonomie: les agents sont des entités avec un savoir faire, un état interne qui auto contrôlent leurs actions.
2. La sociabilité: les agents se communiquent via un langage de communication.

3. La réactivité: les agents ont une perception de leurs environnements et réagissent en temps réel aux changements éventuels de ces environnements.
4. La pro activité: Les agents et tout le long de leurs cycles de vie prennent de l'initiative pour atteindre leurs objectifs.

Par opposition à la rigueur d'un concept mathématique, la notion d'agent chez Russel et Norvig [48] dénote un outil destiné à l'analyse des systèmes. Selon le même avis, cette notion ne doit pas dénoter une caractérisation absolue subdivisant le monde en agents et leur négation.

La proposition de définition mathématique d'un agent autonome par Stan Franklin et al, tente de capturer l'essence d'être un agent, sans perdre en généralité, et se plonger dans les spécificités, et cela afin d'en compenser une classe plus large d'agents. Faut-il en rajouter d'autres restrictions afin de définir des classes particulières d'agents? Dans le cas idéal, ceci, produirait une nomenclature d'agents utilisée sans ambigüité par les chercheurs. Stan Franklin et al, définissent de manière formelle un agent autonome comme ''un système à l'intérieur ou en fait partie d'un environnement qu'il perçoit et agit sur celui-ci, au fil du temps, selon son propre agenda, quant à influencer sur ses perceptions futures''.

Cette définition répond à la norme d'une classe variée et large d'agents. Sans nul doute de dimension bien grande; elle est sans intérêt particulier en l'absence d'un raffinement additionnel. Les sous-classes les plus prometteuses des agents, sont répertoriées dans la table suivante:

Table 1 : Agent Classification

Propriété	Synonyme	Signification
Réactivité	Percevoir et Agir	Répond en temps réel aux changements de l'environnement
Autonomie		Exerce le contrôle sur ses propres actions
Dirigé par le But	Proactif Axé sur le but	Ne réagit pas en réponse à l'environnement uniquement
Continuité		Un processus daemon
Communication	Sociable	Communiqué avec d'autres agents et des humains
Apprentissage	Adaptif	Change de comportement selon son expérience précédente
Mobilité		Capable de migrer d'une machine à une autre
Flexibilité		Ses actions ne sont pas décrites sous forme d'un Script
Caractère		À une personnalité et un état émotionnel

Il est facile de constater, qu'une classification hiérarchique à base d'inclusion des ensembles cités ci-dessus est intrinsèque. Par conséquent, les agents mobiles d'apprentissage sont une sous-classe des agents mobiles.

Notons enfin, que d'autres schémas de classification d'agents sont tout à fait possibles. Nous pouvons classifier les agents logiciels selon leurs tâches, par exemple, les agents du data mining, de filtrage du courriel, ou encore du e-commerce.

De ces différentes schémas de classification des agents, nous retiendrons celle pertinente pour notre thèse et définissant un agent autonome comme une entité logicielle autosuffisante avec un savoir faire et un état interne (ressources), agissant au profit d'une tierce personne ou application et communiquant avec d'autres agents afin de réaliser la tâche pour laquelle il a été crée [49]. Les agents mobiles sont alors définis comme des agents ayant la capacité de se déplacer de site en site de manière proactive, i.e. décident eux mêmes de l'endroit et du moment du déplacement.

3. Mobilité

Les systèmes actuels offrent deux formes de mobilité :

1. **Mobilité au sens faible** qui permet un transfert de code entre différents systèmes. Le code ainsi transféré peut être accompagné d'une partie d'initialisation de données, mais il n'y a pas de migration de l'état d'exécution (pile, variables non statiques).
2. **Mobilité au sens fort** où le système permet la migration du code et de l'état d'exécution d'un programme ou agent vers un nouvel hôte.

La mobilité forte est supportée par deux mécanismes : migration et clonage à distance. Dans la migration, une unité d'exécution est suspendue, puis transmise à sa destination où elle reprend son exécution. La migration est soit proactive ou réactive. Lorsqu'elle est proactive, la destination ainsi que le moment de la migration sont décidés par l'entité elle-même. Dans le cas réactif, la migration est ordonnée par une deuxième entité d'exécution en relation. Le clonage à distance diffère du mécanisme de migration car l'entité d'exécution est cependant toujours attachée à son environnement d'exécution. Le clonage à distance comme la migration peut être proactif ou réactif.

4. Défis des Agents Mobiles [50]:

L'objectif principal de la migration des processus et des objets est de supporter un équilibre de la charge et une optimisation des performances. Les systèmes à code mobile sont destinés, à priori, pour des exigences et à des besoins plus larges. Dans ce contexte particulier, les principaux aspects sont : (1) la personnalisation de service, (2) l'extension dynamique des fonctionnalités d'une application, (3) l'autonomie, (4) la tolérance aux pannes, et (5) de permettre des opérations en mode déconnecté.

Ces aspects tentent de répondre aux problèmes soulevés par le phénomène accablant de pervasivité et d'ubiquité des réseaux d'aujourd'hui. Par pervasive, on entend que la connectivité du réseau est d'accessibilité facile. Par ubiquité, en relate le fait d'exploiter la connectivité du réseau indépendamment de l'endroit physique d'un utilisateur.

Un deuxième phénomène aussi important est la disponibilité forte de technologies facile à utiliser, même par les utilisateurs les plus naïfs (e.g, World Wide Web). Ces technologies ont engendré de nouveaux domaines d'applications ainsi que de nouveaux marchés. Ceci est entrain de basculer la nature du rôle des réseaux, et particulièrement celui de l'Internet. Ils ne peuvent plus être considérés comme de simples technologies de communication. Aujourd'hui, les réseaux modernes d'ordinateurs constituent une innovation dans le support de nouvelles formes de coopération et de communication entre utilisateurs. Le terme tel ''E-Commerce'' et ''Internet phone'' sont symptomatiques de ce changement.

Cependant, cette évolution rencontre des problèmes, et plusieurs défis sont à relever. La croissance de la taille des réseaux pose un problème de passage à l'échelle. La capacité des nœuds d'un réseau de se connecter au réseau d'endroits physiques différents et de manière non continue, engendre une topologie dynamiquement variable du réseau. Un autre sujet aussi pertinent est la diffusion de services et d'applications à un public large de la société. La personnalisation de services, afin d'accommoder la fonctionnalité, ainsi que l'adaptation de l'interface de ces services aux besoins spécifiques et de préférences des différentes classes utilisateurs est une nécessité absolue. Enfin, la nature dynamique de l'infrastructure de communication sous jacente et les exigences du marché implique une flexibilité et une extensibilité des plus accrues.

Beaucoup de chercheurs croient que la technique du code mobile et plus précisément, le paradigme de l'agent mobile, constitue la réponse la plus patente à ce problème de multiple faces.

Le paradigme agent mobile a des caractéristiques intéressantes:

1. L'exécution asynchrone,
2. L'autonomie
3. La reduction de la charge du réseau
4. La tolerance aux panes

Par exécution asynchrone on entend, le franchissement de l'application génératrice de l'agent mobile, de la contrainte d'attendre le résultat, tout en étant dans un état de blocage. Par autonome, on relate le fait que l'agent mobile est en réalité une entité autosuffisante, et ne communique avec l'environnement que par nécessité. L'agent mobile réduit considérablement le trafic dans le réseau car, en principe de par sa vocation, il ne communique qu'avec son site de son exécution courante. Enfin, on dit souvent, qu'un agent mobile est immunisé contre les pannes des sites ou d'un partitionnement du réseau s'il se déplace ailleurs et avant l'arrivée de ces pannes.

En effet, ces quatre caractéristiques constituent la solution aux préoccupations majeures soulevées, dans les réseaux d'ordinateurs modernes, et surtout à une échelle d'Internet. Cependant, le grand défi face à un déploiement et une utilisation massif du paradigme agent mobile au niveau d'Internet est celui de l'aspect sécuritaire. Pour une exécution sûre des agents mobiles, nous devons en disposer d'une infrastructure sous jacente sécuritaire. En général, dans ce cas de la mobilité forte, les chercheurs identifient deux types de menace: les attaques d'un agent contre la plate-forme, et réciproquement les attaques de la plate-forme contre un agent. On peut imaginer plusieurs autres attaques telles que, celles d'un agent contre un autre agent ou d'une plate-forme contre une autre plate-forme, etc. Seules les deux premières catégories d'attaques sont développées dans cette thèse car notre préoccupation majeure est la protection des agents mobiles contre les attaques d'hôtes dits malicieux.

Selon Micheal S. Greenberg et al [51], les agents mobiles et les hôtes sont potentiellement capables de s'en utiliser les uns et les autres de manière abusive. Ils expliquent que ces types d'attaques sont souvent répertoriées selon: (1) les dégâts, (2) le déni de service, (3) le harcèlement, (4) la violation de la confidentialité, (5) l'ingénierie sociale, (6) et les formes complexes telles, celles dirigées par événements et celles d'une composition de combinaisons. Ces deux classes d'attaques sont développées en ordre chronologique dans les deux sections suivantes.

5. Attaques agent contre plate-forme

Les attaques d'agents envers les plates-formes peuvent prendre différentes formes. On distingue principalement sept types d'attaques :

1. la mascarade
2. le déni de service
3. l'accès non autorisé, le vol d'information
4. les dégâts
5. le harcèlement
6. l'ingénierie sociale
7. la bombe logique.

Une ***mascarade*** a lieu quand un agent non autorisé prétend être un autre agent. Elle peut avoir pour but d'obtenir des ressources auxquelles l'attaquant n'a normalement pas accès ou de faire endosser la responsabilité de certaines actions à un autre agent.

Un ***déni de service*** a lieu lorsqu'un agent consomme trop de ressources **(CPU, Bande Passante de la Connexion Réseau)**. Ces attaques peuvent être intentionnelles ou non (i.e., erreur de programmation). Le problème avec un agent mobile est que le code est en principe écrit en dehors de la plate-forme qui l'exécute et il peut contenir du code malveillant. Quand un agent arrive sur une plate-forme, il faut qu'il soit authentifié puis soumis à la politique de sécurité le concernant. Cela permet d'empêcher l'accès d'utilisateurs ou de processus non autorisés à des ressources protégées. S'ils arrivent à y avoir accès il y aura **vol d'information**. Des dégâts

peuvent être causés lorsqu'un agent détruit ou modifie des ressources ou des services en les reconfigurant, en les modifiant ou en les effaçant de la mémoire ou du disque. Tous les autres agents sur la plate-forme qui utilisent ce service ou cette ressource sont touchés.

Il y a une attaque de type **harcèlement** lorsqu'un agent mobile ennuie les gens par des attaques répétées : cela arrive lorsqu'un agent mobile montre à répétition des images publicitaires non demandées à l'écran de l'hôte sur lequel il s'exécute.

L'**ingénierie sociale** a lieu lorsque les personnes ou les hôtes sont manipulés par l'agent mobile en utilisant la désinformation ou la coercition. Par exemple, un agent mobile peut demander le mot de passe de l'utilisateur sous la fausse autorité de l'administrateur système.

Une **bombe logique** est une des attaques dont le déclenchement est basé sur un événement externe (une date, un emplacement de l'hôte dans un certain périmètre du réseau).

6. Attaques plate-forme contre agent

On distingue cinq types d'attaques de la plate-forme envers les agents :

1. la mascarade
2. le déni de service
3. le vol d'information
4. l'altération
5. la bombe logique.

Une **mascarade** a lieu lorsqu'une plate-forme se fait passer pour une autre plateforme dans le but de tromper l'agent par rapport à sa destination normale et de le faire quitter son domaine de sécurité. Cela peut permettre à la plate-forme d'obtenir de l'information sensible contenue dans l'agent. Cela fait du dommage à l'agent et à la plate-forme dont l'identité a été abusée.

Lors d'un **déni de service**, la plate-forme refuse d'effectuer les services demandés par l'agent. Ainsi, elle peut ignorer les demandes de service, introduire des délais inacceptables

pour des tâches critiques, ne pas exécuter le code de l'agent ou bien le terminer sans avertissement. D'autres agents qui attendent la réponse de cet agent seront en inter blocage (**deadlock**). Par exemple, si une plate-forme ne donne pas d'accès réseau à un agent mobile, il ne parvient pas à se déplacer.

Le **vol d'information** est un danger très important car la plate-forme a accès au code, à l'état et aux variables de l'agent mobile; il est donc dangereux de faire exécuter des algorithmes propriétaires ou de stocker des informations sensibles (clé privée du propriétaire) dans l'agent. L'encryptage classique n'est valable ici que pour le code et les données qui ne seront pas utilisés sur cette plate-forme. En outre, même si la plateforme n'est pas capable d'obtenir directement de l'information par l'agent, elle peut en déduire par les services demandés, la destination de l'agent ou encore à partir des agents avec qui il communique. Ainsi, si l'agent mobile communique avec un fournisseur d'un service, la plate-forme pourra avertir un autre fournisseur concurrent qui démarchera l'agent. Un agent se déplace sur plusieurs plates-formes, dans des domaines de sécurité différents; la plate-forme a accès au code, à l'état et à l'état d'exécution de l'agent mobile. En conséquence, il est possible qu'une des plates-formes modifie le code, l'état ou l'état d'exécution : cela constitue une **altération** de l'agent. La plate-forme peut aussi modifier la communication de l'agent mobile. Cela serait désastreux dans le cas de transactions financières. Un agent prend des risques à chaque fois qu'il se déplace sur une nouvelle plate-forme. De plus, il peut devenir impossible de retrouver la plate-forme responsable d'une altération si elle n'est pas détectée tout de suite. On peut distinguer deux cas : celui où l'agent mobile se déplace depuis sa plate-forme mère vers une autre puis revient, il est appelé ‘’single hop problem ‘’; celui où plus d'une plate-forme étrangère sont visitées, il est appelé ‘’ multi-hop problem ‘’ Bien sûr, les risques sont plus importants et plus difficiles à gérer dans le dernier cas.

De même que pour les bombes logiques d'un agent contre une plate-forme, une bombe logique d'une plate-forme contre un agent est déclenchée par un événement extérieur. Ainsi, une plate-forme peut attaquer un agent à cause de l'identité de son propriétaire ou des données qu'il transporte.

Les prochaines sections de ce chapitre sont consacrées aux techniques de la littérature pour protéger la plate-forme et celles pour protéger l'agent.

7. Protection de la plate-forme

Beaucoup de techniques utilisées pour protéger la plate-forme de l'agent sont proches de celles utilisées dans une architecture distribuée classique (exemple : client/serveur). D'autres sont tirées de techniques de sécurité du code mobile et du contenu exécutable. Contrairement aux premières, ces dernières n'ont pas encore été suffisamment testées. Le concept d'agent mobile pose des problèmes qui n'ont pas été rencontrés auparavant. Greenberg et al identifient les techniques de protection d'hôtes suivantes: (1) authentification de certificat, (2) supervision et contrôle du niveau d'accès, (3) vérification du code, (4) limitation de temps, (5) limitation de domaine, i.e. de destination, (6) limitation de duplication, (7) et le journal d'audit.

Dans sa recherche des éléments qui font la différence entre la sécurité des systèmes conventionnels de ceux des agents mobiles, Chess[20, 2] a identifié quatre hypothèses que les agents mobiles ne respectent pas **:**

1. lorsqu'un programme effectue une action sur un système conventionnel, il est facile d'identifier la personne à qui cette action peut être attribuée ou qui a effectué cette action;
2. tous les logiciels proviennent de sources facilement identifiables et généralement de confiance;
3. la plupart des menaces proviennent d'attaquants qui exécutent des programmes dans le but d'obtenir des résultats non autorisés;
4. les programmes ne se transmettent que si les personnes le font intentionnellement.

Le but de la protection de la plate-forme est d'empêcher l'agent d'interférer avec celle-ci. La plupart des techniques associées à cette protection sont basées sur le modèle de forteresse, selon lequel, on cherche à protéger l'hôte en ayant un système fermé qui n'est accessible que par des interfaces bien définies. Dans ce contexte, on utilise le concept de ***moniteur de référence*** qui est un programme contrôlant l'accès de l'agent mobile aux informations, ressources et services de la plate-forme. Ce moniteur de référence applique un certain nombre de règles pour l'accès de l'agent à ces ressources. Ces règles sont regroupées dans la politique de sécurité de la plate-forme.

Les accès aux ressources et aux services de la plate-forme sont toujours assujettis à un contrôle à priori du moniteur de référence (il est impossible d'obtenir directement une ressource). Le moniteur de référence doit être résistant aux altérations et suffisamment petit pour qu'il puisse être analysé et testé [22]. Les techniques de protection de la plate-forme sont :

1. l'authentification des agents
2. le carré de sable,
3. le contrôle d'accès,
4. la vérification du code,
5. l'estimation de l'état
6. l'historique des hôtes
7. le code avec preuve
8. les techniques de limitations
9. le journal d'audit.

7.1 Authentification des agents

Le problème de l'authentification des agents mobiles est similaire à celui qui se pose en milieu distribué. Ici, on a deux buts : vérifier l'intégrité du code et authentifier ses auteurs et utilisateurs. Greenberg et Byington [23] proposent de signer l'agent mobile numériquement avec un algorithme cryptographique à clé publique. Cet algorithme peut soit générer un certificat qui assure l'intégrité du code et qui permet d'authentifier le propriétaire et le producteur de l'agent mobile pour l'hôte, soit crypter l'agent de sorte que seul le réceptionnaire puisse le décoder. Cela assure aussi la confidentialité de l'agent mobile. Berkovits, Guttman et Swarup [24] proposent un modèle pour l'authentification des agents mobiles. Ils distinguent cinq sujets : l'auteur de l'agent, le programme (écrit et signé par l'auteur), l'expéditeur de l'agent, l'agent (constitué des données et du programme) et les hôtes où il a été exécuté. II découpe la vie de l'agent en trois périodes : la création du programme, la création de l'agent et la migration de l'agent.

La création du programme consiste en l'assemblage du code et d'une fonction d'estimation de l'état **(State Appraisal Function)**. Cette fonction donne le nombre maximum

de permission, qui doit être accordé à l'agent en fonction de son état. Le tout est signé par les auteurs et éventuellement par un organisme de vérification du code. Pour ce qui est de la création de l'agent, l'expéditeur rajoute une autre fonction d'estimation de l'état qui calcule les permissions dont l'agent mobile a besoin pour s'exécuter en fonction de son état. Cela est signé par le créateur. En ce qui concerne la migration de l'agent, il en existe plusieurs types qui sont basés sur des relations de confiance différentes :

1. le changement d'hôte où l'hôte **I1** envoie l'agent vers un autre hôte **I2;**
2. la délégation où l'hôte **I1** envoie l'agent vers un autre hôte **I2** où il sera exécuté sous la responsabilité de l'expéditeur **I1;**
3. le changement d'agent quand un agent décide lui-même de migrer, il s'exécute alors sous la responsabilité de son créateur;
4. la délégation d'agent où l'agent se délègue à un hôte.

Quand un hôte reçoit une demande pour exécuter un agent, il vérifie la signature de l'auteur du programme et la signature de l'expéditeur, puis, il authentifie l'agent comme admissible en vérifiant un certain nombre de théorèmes.

Chess [20] évoque les limites de cette approche qui sont les mêmes que dans une architecture répartie, mais avec encore plus de problèmes. La première limite est bien sûr la distribution des clés publiques car les solutions qui fonctionnent avec un nombre réduit d'utilisateurs ou qui utilisent occasionnellement la cryptographie ne sont pas adaptées pour les agents mobiles. Il est important d'avoir un moyen d'obtenir facilement le certificat de la clé publique d'un sujet.

La seconde limite, aussi décrite par Greenberg et al. [23]**,** est le type de certification; on sait qui est l'auteur du programme, mais cela ne veut pas dire que le programme est inoffensif. Pour éviter cela, il faut que des autorités certifient ce caractère inoffensif. Jansen et Kaygannis [22] ajoutent qu'il est parfois difficile d'avoir des vérifications de qualité en peu de temps. Le problème devient alors celui d'une certification de l'état de l'agent mobile car il a été exécuté sur des hôtes potentiellement dangereux.

7.2 Carré de sable

La technique du carré de sable (SandBox) consiste à isoler un programme dans son propre espace de fautes de façon logicielle. L'environnement est restreint en termes de privilèges et de ressources. Le code est exécuté dans une sorte de ‘’carré de sable‘’. Un agent ne pourra pas modifier la plate-forme, ni les autres agents qui s’exécutent sur la plate-forme. Par exemple, une applet java ne peut pas accéder au système de fichier local.

7.3 Contrôle d'accès et d'autorisation

La technique du contrôle d'accès et d'autorisation se rapproche du carré de sable**.** Le moniteur de référence donne des autorisations à un agent mobile pour un certain nombre de ressources en fonction du résultat de son authentification. Pour savoir quelles autorisations donner à quel agent, le moniteur de référence applique les règles de la politique de sécurité. Ces règles touchent toutes les ressources qu'un agent peut en avoir besoin : accès au réseau, aux disques, etc. Dans le langage Java [28], le moniteur de référence est le ‘’security manager’’, il prend en charge tous les accès aux ressources. Par exemple, un agent mobile qui tente de lire un fichier dont il ne détient pas les droits de lecture est arrêté par le moniteur de référence. Cette technique est différente du carré de sable car elle permet, d'une part d'avoir une granularité plus fine sur la protection et l'autorisation et d'autre part, une adaptation des droits d'accès en fonction des agents. Le problème du carré de sable ne se pose plus, car une application avec des besoins forts ou particuliers en ressources est apte à les avoir si la politique de sécurité le permet.

7.4 Vérification du code

Le but de cette technique est de vérifier si le code d'un agent mobile est un programme valide. Si un vérificateur de code trouve des opérations illégales dans un agent, celui-ci ne sera pas exécuté. Un exemple de refus pourrait être un programme qui exécute une chaîne de caractères aléatoire comme un segment de programme. Dans les langages interprétés, cette technique est parfois nommée « interprétation du code inoffensif ». Dans Java, il existe un vérificateur de byte-code qui effectue une vérification sur les types, il fait aussi des vérifications pendant l'exécution [28].

7.5 Estimation de l'état

La technique d'estimation de l'état ‘’**State Appraisal**’’ est utilisée pour protéger une plate-forme par rapport à un agent dont l'état aurait été modifié par une plate-forme défaillante ou ennemie. A ce titre, elle pourrait aussi apparaître dans les techniques de protection de l'agent, car elle permet de détecter une altération de son état. L'auteur et l'utilisateur de l'agent seront protégés contre une utilisation frauduleuse de leur agent. **Farmer**, **Guttman** et **Swarup** [24] ont proposé l'idée de la fonction d'estimation de l'état qui prend comme entrée l'état de l'agent et qui donne en sortie un ensemble de ressources dont l'agent a besoin pour exécuter sa tâche. Le producteur de l'agent et l'utilisateur créent leurs propres fonctions. Chacune est signée numériquement pour empêcher toute modification liée à l'agent. L'auteur écrit une fonction **max** qui donne le maximum de ressources que l'agent peut demander. L'utilisateur de l'agent écrit une fonction **req** qui donne les ressources que demande l'agent; cette fonction peut minimiser le coût que l'utilisateur doit payer pour les ressources de son agent. S'il n'y a pas de modifications de l'état de l'agent, on aura **req(R)** inclus dans **max(R)**. Si jamais ce n'est pas le cas, c'est qu'il y a eu modification de l'état de l'agent mobile. Ces fonctions permettent aussi à la plate-forme de savoir de quelles ressources l'agent a besoin et de les lui allouer si cela est conforme à sa politique de sécurité. Cependant, la manière dont cette théorie sera mise en pratique n'est pas claire car l'ensemble des états possibles d'un agent est très grand, et écrire des fonctions d'estimation de l'état sera certainement facile pour détecter les attaques évidentes; pour des attaques plus subtiles, cela devient plus compliqué. De plus, il n'est pas toujours possible de distinguer un résultat normal d'une tentative de tromperie.

7.6 Historique des hôtes

Westhoff et al. [29] et surtout **Ordille** [25] ont proposé l'idée d'évaluer la confiance qu'a un hôte dans un agent à partir de son identité et des plates-formes sur lesquelles il s'est exécuté antérieurement. Pour cela, il est nécessaire de mettre à jour un journal avec l'identité de toutes les plates-formes qui ont été visitées par l'agent. Ce journal est protégé crypto graphiquement pour éviter toutes tentatives de modifications : à chaque fois qu'un agent mobile migre de la plate-forme sur laquelle il se trouve, il signe le journal qui indique son identité et celle de la destination de cet agent mobile. Lorsqu'un agent arrive sur une nouvelle plate-forme, cette dernière décide si elle l'exécute ou non et quelles ressources elle lui accorde

en fonction de l'historique des hôtes visitées par l'agent. Cette technique est efficace dans le sens où, avec la signature numérique, une entrée dans l'historique est non répudiable. On peut donc avoir confiance dans l'historique. Le problème de cette technique est que lorsque le nombre d'hôtes visités augmente, l'historique peut prendre un poids prohibitif. De plus, il n'existe pas encore à notre connaissance d'algorithmes pour évaluer l'historique et en déduire le niveau de confiance et de risque.

7.7 Code avec preuve

Necula et **Lee** [26] ont proposé le code avec preuve ou **PCC** ‘’Proof Carrying Code’’ qui permet à un système de déterminer de manière certaine et instantanée si le code qu'il a reçu d'un autre système est sûr à installer et à exécuter. L'idée est que le producteur du code donne une preuve encodée, que le code adhère à la politique de sécurité du consommateur. La preuve est encodée de telle sorte qu'elle puisse être transmise numériquement au consommateur et qu'elle puisse être validée rapidement avec un procédé fiable, simple et automatique. Une fois que le code a été vérifié, il peut être exécuté sans risque. Le **PCC** se fait en deux étapes. A la première, le consommateur reçoit le code non vérifié et donne un prédicat sur ce code (de type logique du premier ordre) qui, s'il est vérifié, veut dire que le programme est conforme à sa politique de sécurité. La deuxième étape est la construction de la preuve. Cette partie est souvent difficile à faire. Puis, la preuve est envoyée au consommateur qui la vérifie, cette phase est rapide à effectuer.

Selon Necula et Lee [26] les points forts de **PCC** sont la généralité (il permet d'avoir des politiques de sûreté plus abstraites et plus adaptables à la plate-forme que la protection de la mémoire), l'automatisme du mécanisme de vérification et la confiance qui en découle, l'efficacité (la vérification est rapide), le fait qu'il n'y ait pas besoin de relation de confiance, et la flexibilité (PCC est indépendant du langage de programmation). Par rapport aux autres techniques de protection de la plate-forme (isolation logicielle, langage interprété), les résultats au niveau temps d'exécution sont plus rapides. De plus, **PCC** permet d'éviter d'exécuter du code qui se termine de façon ‘’brutale’’ : si on utilise une technique de protection de la plate-forme qui demande une vérification en temps réel du code comme le vérificateur de code Java, le code sera exécuté et se terminera ‘’brutalement’’. Cela peut prendre du temps à la plate-forme pour reprendre une exécution normale. Le **PCC** permet donc de gagner du temps **CPU** en n'exécutant pas cet agent. Enfin, cette technique est

préventive, alors que signer le code permet d'identifier et d'authentifier le code mais ne permet pas d'éviter l'exécution de code dangereux.

Cependant, un de ses problèmes réside dans le dilemme entre l'optimisation du code et la difficulté à construire la preuve, car plus le code est optimisé, plus le prédicat est difficile à prouver. Un autre problème vient des contraintes de temps car, pour pouvoir exercer un contrôle du nombre d'instructions, l'implémentation de l'agent doit rajouter un compteur d'instructions. Une voie de recherche est de concevoir un outil qui rajoute automatiquement ces instructions. L'automatisation des **PCC** est aussi une voie active de recherche. Les résultats des expériences montrent que la taille des PCC est importante et augmente parfois exponentiellement avec le code. Des améliorations sont donc possible. Des recherches peuvent être faites sur d'autres propriétés que le **PCC** pourrait prouver.

7.8 Techniques de limitations

Greenberg et Byington [27] proposent trois types de limitations sur l'agent qui peuvent aider à protéger un hôte. Ces limitations doivent être choisies en fonction de la tâche de l'agent car, le cas échéant, elles peuvent l'empêcher de l'effectuer :

1. Limitation en temps : L'agent mobile n'a le droit de rester dans la couche d'exécution qu'un certain temps (relatif ou absolu). Une fois le temps écoulé, il est détruit ou revient à sa plate-forme de départ.
2. Limitation en destination : le nombre de destinations possibles de l'agent mobile est limité. De plus, les migrations ne pourront se faire que vers une liste d'hôtes définie dès la création de l'agent ou dans une certaine partie du réseau. Par exemple, l'agent ne pourra aller que sur les plates-formes d'une certaine organisation.
3. Limitation de duplication : Un agent ne peut être autorisé à migrer qu'un certain nombre de fois.

7.9 Journal d'audit

Le journal d'audit enregistre toutes les actions de l'agent. Ainsi, après la découverte d'une action frauduleuse, un agent fraudeur pourra être identifié. Cela peut être utile pour des poursuites judiciaires.

Après avoir décrit les techniques pour protéger les plates-formes, nous présentons les techniques proposées par les chercheurs pour protéger les agents contre les attaques de plates-formes malicieuses. Toujours et selon Greenberg et al, ils existent deux catégories de techniques de protection des agents mobiles : (1) de tolérance aux fautes, (2) et de cryptage. Dans l'objectif d'étoffer le sujet nous avons inclus d'autres techniques disponibles dans la littérature, surtout celles qui constituent une assise au travail de recherche de notre thèse.

8. Protection des agents

Alors que les contre-mesures visant à protéger les plates-formes sont largement inspirées des systèmes conventionnels en employant des méthodes préventives, les techniques de protection des agents font plutôt de la détection afin d'accroitre la survivance de leurs propriétaires [11]. Cela est dû au fait que l'agent est complètement dépendant de la plate-forme sur laquelle il s'exécute et ne peut, par lui-même empêcher une attaque. Par contre, il est possible de la détecter ou de la rendre moins dangereuse. Le principal problème vient des données et des informations d'état. En effet, supposons que l'utilisateur de l'agent signe l'agent (code + données + informations d'état) et l'envoie sur la première plate-forme. Lors de la prochaine migration. Il n'y aura plus de vérification possible sur l'état et les données car la plate-forme les aura modifiés et elle est peut être ennemie. Il est possible d'avoir du code signé. Cependant, on ne peut pas protéger les parties variables de l'agent. La protection des agents contre la plate-forme est un domaine de recherche récent, subtil et très important car les problèmes qui se posent sont différents de ceux rencontrés dans les systèmes traditionnels. Parmi les techniques de protection, on trouve :

1. l'enregistrement d'itinéraires avec des agents coopérants,
2. les traces cryptographiques,
3. le calcul de fonctions cryptographiques,
4. la boite noire limitée dans le temps,
5. la génération de clés à partir de l'environnement,
6. la protection de l'agent par tolérance aux fautes,
7. l'encapsulation des résultats partiels,
8. Tamper free hardware,

9. le masquage du code,
10. estimation de l'état,
11. limitation du temps d'exécution.

8.1 Enregistrement d'itinéraires avec des agents coopérants

Roth [28] propose de donner une tâche à deux agents qui vont parcourir un ensemble de plates-formes et qui ont des itinéraires disjoints, un des deux agents mobiles enregistrant l'itinéraire de l'autre. Avant de migrer, un agent envoie à l'autre agent, à travers un canal authentifié, l'identité de la dernière plate-forme, celle de la plate-forme actuelle, ainsi que celle de la plate-forme où il va. L'autre agent maintient un journal de l'itinéraire et vérifie qu'il n'y a pas de contradiction.

Roth [28] fait plusieurs hypothèses. La première est qu'aucun hôte sur l'itinéraire d'un agent ne coopère avec un hôte de l'itinéraire de l'autre agent. Il suppose aussi l'existence d'un canal de communications. Enfin, il suppose que les identités sont données sans altération à l'agent. Toutefois, la méthode recèle un certain nombre d'inconvénients. En effet, mettre en place le canal de communications à chaque migration est une opération coûteuse. Si un agent est tué, le protocole n'est pas capable de savoir lequel des deux hôtes est responsable. Si une plate-forme déclare avoir reçu un agent d'une plate-forme et que cela n'est pas vrai, le protocole ne peut décider qui est coupable. Enfin, si un hôte reçoit deux agents qui viennent du même hôte, il est possible d'inter changer les agents qui enregistrent 1' itinéraire des deux autres.

Roth [29] adapte un protocole de paiement basé sur le principe de deux agents coopérants. Ce protocole peut être généralisé à plus de deux agents. Pour certaines applications, il est possible qu'un agent soit statique sur la plate-forme de l'utilisateur.

8.2 Traces cryptographiques

Vigna [14] propose une technique qui permet de détecter toute exécution anormale d'un agent sur une plate-forme à partir d'un fichier, une sorte de ‘’résumé de l'exécution’’ appelé trace. Ces traces sont signées par chaque plate-forme, ce qui permet de retrouver la plate-forme coupable d'une mauvaise exécution de l'agent. A la fin de l'exécution de l'agent, si

l'utilisateur de l'agent a des suspicions quant à la bonne exécution de l'agent, il ré-exécute l'agent grâce aux traces et, en cas d'erreur, il peut ainsi retracer la plate-forme coupable. Une trace est une paire (n,s) où n désigne un identifiant unique d'une ligne de code et s la signature de cette ligne de code. Pour les opérations qui ne modifient les variables que par rapport à d'autres variables internes (opérations blanches) la signature est vide. Le Tableau 2.1 illustre la notion de trace.

Code	Trace
1. read(x)	(1, x=8)
2. y=x+z	(2, -)
3. m=y+1	(3, -)
4. K=cryptInput	(4, k=5)
5. M=m+k	(5, -)

Table 2: Example of a Trace

Après exécution, une plate-forme crée un fichier en appliquant une fonction de hachage aux traces de l'exécution. Ce fichier est ensuite signé par la plate-forme grâce à sa clé privée. Puis, elle envoie ce fichier au prochain hôte de l'agent avec le code et l'état de l'agent. Une fois que l'agent a terminé son exécution et est rentré sur la plateforme de son propriétaire, ce dernier peut décider s'il veut vérifier l'exécution de l'agent ou non. Si c'est le cas, il ré-exécute l'agent à partir de son état initial, puis compare le résultat de la fonction de hachage de ses traces avec celui de l'exécution à distance. S'ils sont égaux, c'est que l'hôte n'a pas triché et on passe à l'hôte suivant.

Cette approche du ‘’Code Trace’’ détecte toutes les attaques qui donnent un état différent, du moment que l'hôte ne ment pas sur les entrées. Cependant, des critiques peuvent être émises. La première est celle de la consommation de ressources : les algorithmes à clé publique sont particulièrement consommateurs de temps C.P.U., la taille des traces peut devenir assez importante (même en appliquant plusieurs mécanismes pour en diminuer la taille). De plus, il était supposé que le code était statique. Or, la plupart des langages de type

Script proposent le code dynamique. Une autre limite est le fait qu'il faut attendre la fin de l'exécution de l'agent pour pouvoir détecter une attaque. Si une attaque doit être détectée avant un certain temps, cela n'est pas possible. Enfin, il a été supposé que les agents ne partageaient pas de mémoire et n'étaient pas en ‘’multithread’’.

Une amélioration à ce modèle pourrait être apportée pour ne plus avoir ces limitations. Hohl [31, 33] propose une adaptation de la technique des traces cryptographiques en faisant la vérification de l'exécution à partir des traces, lors de l'arrivée d'un agent mobile sur une nouvelle plate-forme. Il a montré grâce à des mesures que la vérification engendrait une consommation du C.P.U. à peu près égale à celle de l'exécution. Cette approche permet d'éliminer un certain nombre de désavantages de la méthode de **Vigna**. En effet, la taille des traces est réduite car on n'échange que les traces d'une exécution d'un hôte et non pas la somme des traces de tous les hôtes visités précédemment; la détection d'une attaque se fait juste après l'attaque. Cependant, des optimisations de la taille des variables d'états (donnée et variables d'exécution) sont possibles.

En effet, après son exécution sur une plate-forme, l'agent va être envoyé sur la prochaine plate-forme avec les variables d'états avant exécution et après exécution, ainsi que les traces. Cela permet à la plate-forme suivante de vérifier la bonne exécution de l'agent sur la plate-forme précédente. L'optimisation de la taille des variables d'états peut se faire avec une méthode qui permettrait de choisir comment envoyer ces deux états (avant et après exécution): soit deux états distincts, soit un état et une différence par rapport à cet état. De plus, une méthode de choix entre une entrée entière fournie par la plate-forme où le résultat de la première opération sur cette entrée est souhaitable. En effet, dans la trace, on conserve normalement le résultat de la première ligne de code qui utilise l'entrée. Mais, si l'entrée est de taille inférieure à ce résultat, il est préférable de la conserver. Enfin, reste le problème des entrées qui peuvent être manipulées par l'hôte sans protection, et celui de deux hôtes ennemis collaborant et qui se suivent dans l'itinéraire de l'hôte.

8.3 Calcul de fonctions cryptographiques

Sander et Tschudin [32] ont proposé une technique pour que l'agent mobile devienne une boite noire pour la plate-forme sur laquelle il s'exécute. Ainsi, ils proposent une méthode

pour que l'agent signe des documents. Il s'agit de faire exécuter à la plate-forme un programme qui contient une fonction cryptée sans qu'elle soit capable de décoder cette fonction.

Le problème est le suivant :

''Alice a un algorithme qui calcule la fonction f. Bob a une entrée x et veut calculer f(x) pour Alice, mais elle ne veut pas que Bob apprenne quoique ce soit de f. De plus, Alice et Bob ne doivent pas s'échanger de message pendant le calcul de f(x)''.

Pour résoudre ce problème, il faut supposer que **A** puisse crypter f, ce qui donne **E(f).** En voici ci-après leur méthode :

1. Alice crypte **f.**
2. Alice crée un programme **P(E(f))** qui implémente **E(F)**.
3. Alice envoie **P(E(f))** à Bob.
4. Bob applique **P(E(f))** à **x.**
5. Bob envoie **P(E(f))(x)** à Alice.
6. Alice décrypte **P(E(f))(x)** et obtient **f(x)**.

Le problème est maintenant de trouver des modèles cryptographiques qui permettent de calculer **E(f).** Sander et Tschudin[32] ont trouvé une méthode pour les fonctions *f* de type polynomial et rationnel pour calculer **E(f).** Cependant, plus de recherche est nécessaire dans ce domaine car tous les programmes ne peuvent être représentés seulement comme une fonction polynomiale ou rationnelle. Cette méthode est utilisable pour l'instant dans un domaine très restreint.

8.4 Boite noire limitée dans le temps

Hohl [33] décrit une méthode de type boîte noire pour protéger un agent contre une plate-forme ennemie. Elle consiste à modifier le code de l'agent de telle sorte qu'il fasse le même travail, mais que la lecture du code ne permette pas de comprendre ce que fait la fonction ou de la modifier sans détection. Le problème est qu'il n'existe pas de solution complète à ce problème. Le calcul de fonctions cryptographiques est une solution mais elle

n'est pas suffisante car elle ne peut pas être appliquée à tous les agents. C'est pourquoi Hohl [33] introduit le concept de boîte noire limitée dans le temps.

Les propriétés sont les mêmes que celles de la boîte noire, mais elles ne sont valables que pour un certain temps qui est lié à l'agent. Dans ce cas, il faut tout d'abord que le code de l'agent change à chaque début de l'intervalle de protection. Il faut aussi des algorithmes de confusion (obfuscating ou mess-up algorithms) qui rendent une analyse du code difficile (donc longue) mais pas impossible pour un attaquant. Hohl décrit les caractéristiques que devront avoir ces algorithmes : l'algorithme devra être paramétrable avec un grand nombre de valeurs possibles, il ne devra pas être possible de casser la protection avant l'exécution. Le premier problème est la quantification de l'intervalle de protection que donne l'algorithme de confusion : comment déterminer l'intervalle à partir de l'algorithme ? Il manque un modèle qui donne le degré de masquage d'une variable et la complexité pour la retrouver. De plus, il faut déterminer à partir de quel intervalle cette protection est utile pour l'agent : les agents ont besoin de plus ou moins de temps pour effectuer leur travail. Donc, si l'intervalle est trop faible, le nombre d'agents où cette protection pourra être utilisée sera réduit.

8.5 Génération de clés à partir de L'environnement

Riordan et Schneier [34] proposent une méthode pour qu'un agent fasse une certaine action quand une condition de l'environnement est vraie. Une clé est générée quand cette condition est vraie. Elle peut être utilisée pour décrypter le code de l'agent, pour décrypter un message destiné à l'agent dont l'expéditeur veut qu'il ne soit déchiffrable que dans certaines conditions. La plate-forme ne doit pas savoir de quoi dépend cette condition car elle maîtrise tout l'environnement. Un ‘’agent sans indice’’ est un agent qui possède du code encrypté et une fonction qui permet de générer la clé de décryptage en fonction de l'environnement. Si l'environnement n'a pas les conditions pour générer la clé, il est impossible de deviner la fonction de l'agent. Il est possible de générer une clé à partir de données fixes sur un canal (ex : serveur de news, pages web). Cependant, l'utilité de cette méthode est fonction des entités qui ont accès directement ou indirectement au canal et de celles qui peuvent manipuler les données et de la manière dont varie l'observation du canal en fonction des observateurs. Il est aussi possible de générer une clé à partir du temps : certaines méthodes permettent de générer la clé seulement avant une certaine date, alors que d'autres ne le permettent qu'après

une date déterminée. En utilisant les deux, on peut arriver à générer une clé seulement pendant un intervalle de temps fixé.

8.6 Protection de l'agent par la tolérance aux fautes (Vote)

Schneider [15] propose une méthode pour qu'un agent mobile soit tolérant aux fautes. Une plate-forme fautive peut se comporter comme une plate-forme ennemie. Donc, en appliquant des méthodes pour se protéger des fautes, on peut aider à protéger l'agent contre les attaques de plates-formes. La méthode décrite consiste à utiliser la duplication et le vote. Plusieurs copies du même agent sont envoyées sur plusieurs plates-formes. Chaque plate-forme de l'étape i prend comme entrée la majorité des entrées qu'elle a reçues des plates-formes de l'étape précédente (i -1). Quand elle termine, elle envoie son résultat à toutes les plates-formes de l'étape (i+l)**.** Il est aussi possible de savoir quelle plate-forme a donné des résultats erronés grâce à un historique des chemins. Cette technique s'applique aux applications où les agents peuvent être dupliqués, où la tâche peut être décomposée sans problème et où la survie est une question importante. Cependant, l'inconvénient majeur est, bien sûr, les ressources additionnelles qui sont consommées par la duplication. Un problème possible est de vérifier qu'un des prédicats ne diverge pas car, dans ce cas, ce prédicat n'enverrait pas de résultats aux prochaines plates-formes. En effet, la technique utilisant les machines à états n'est pas applicable ici car les agents dupliqués peuvent exécuter des requêtes différentes ou des requêtes dans un ordre différent.

8.7 Encapsulation des résultats partiels

Cette méthode consiste à détecter une altération de l'agent en encapsulant les résultats des calculs de l'agent à chaque plate-forme visitée. La vérification de la validité de ces résultats se fait sur la plate-forme du propriétaire ou sur un intermédiaire placé sur la trajectoire de l'agent. L'encapsulation a plusieurs buts comme la confidentialité (assurée par la cryptographie), l'intégrité et la non-répudiation (assurées par une signature numérique). Les informations qui sont encapsulées dépendent de l'agent. Il y a trois manières d'encapsuler les résultats : avec l'agent, avec la plate-forme ou avec une troisième partie de confiance. La dernière solution n'a pas été véritablement explorée, mais a seulement été évoquée par Yee [55]. Young et Yung [36] qui ont proposé une méthode cryptographique basée sur la

cryptographie à clé publique permettant d'encrypter de petits volumes de données en résultats. L'intérêt est que le volume des données encryptées est petit. Cela s'adapte bien au cas des agents mobiles car les résultats partiels sont petits et, avec sa capacité de stockage limitée, l'agent ne peut pas transporter de grands volumes de données cryptées. Les deux auteurs ont montré non formellement, comment modifier ce schéma d'encryptage pour que la corrélation entre des données encryptées soit plus difficilement calculable. Cela permet de rendre les agents moins traçables.

Yee [55] propose d'encapsuler les résultats dans un PRAC (code d'authentification des résultats partiels). Avant de migrer vers une autre plate-forme, l'agent calcule un « résumé » (le **PRAC)** des résultats partiels. La clé utilisée est choisie parmi une liste de clés secrètes que l'agent et son propriétaire connaissent. Une fois le calcul effectué, la clé est effacée. Un calcul de code d'intégrité de message est effectué pour assurer l'intégrité des résultats partiels. L'agent migre alors vers la nouvelle plateforme en ayant avec lui le résultat partiel et le PRAC. Si une des plates-formes suivante est ennemie, elle ne pourra pas modifier le résultat partiel sans détection car elle ne pourra pas calculer le **PRAC** : elle n'a pas la clé associée.

Jansen et Kaygannis [22] ont mis en évidence plusieurs limitations à cette technique. En effet, si une plate-forme conserve l'ensemble de clés ou la fonction génératrice de clé et que l'agent revisite cette plate-forme ou une plate-forme alliée à cette dernière, un résultat partiel pourra être modifié sans possibilité de détection. De plus, la méthode ne permet pas d'assurer la confidentialité des résultats. Pour y remédier, on peut ajouter une étape de cryptage des données.

Roth [28, 29] présente une méthode orientée plate-forme qui est une amélioration du **PRAC.** Chaque plate-forme rajoute ses résultats à la chaîne de résultats en les liants aux précédents : une plate-forme signe numériquement ses résultats avec sa clé privée, et applique une fonction de hachage à la concaténation de la chaîne et de ses résultats cryptés. Cette méthode assure l'intégrité en cryptant chaque résultat avec la clé du propriétaire de l'agent. Cette intégrité est plus forte que le **PRAC** car il est impossible pour une plate-forme de modifier son entrée propre si elle est revisitée par l'agent mobile, ou pour une plate-forme collaboratrice.

8.8 Tamper Free Hardware

Wilhem et al. [40] étudie la notion de confiance dans le cadre des agents mobiles. La confiance dans un tiers est définie comme la croyance qu'il va respecter sa politique de sécurité. Les auteurs discernent deux types d'approche : une optimiste et une pessimiste. Dans l'optimiste, on fait preuve de confiance à l'autre entité et on essaye de détecter les violations de politique. Si c'est le cas, il y a une sentence. On distingue alors deux types de confiance : celle basée sur la réputation et celle basée sur une sentence explicite. Dans l'approche pessimiste, on prévient les violations de la politique de sécurité. L'approche de Wilhem et al. [40] consiste à déléguer la confiance dans un matériel résistant aux altérations, qui fournit à l'agent mobile un environnement d'exécution sûr**.** Cela permet aux données d'être protégées de manière sûre. Le problème est la protection de cet environnement car tout système est faillible pourvu que l'attaquant ait suffisamment de ressources et de temps.

8.9 Masquage du code

Bazzi [38] propose une définition pour le masquage du code et pour la résistance aux modifications. Il décrit sa méthode qui consiste à rajouter du code de manière systématique et automatique en gardant la même fonction. Une solution pour les fonctions polynomiales est proposée, le masquage et la résistance aux modifications sont prouvés. Le problème du masquage du code pourrait être mieux spécifié en tenant compte notamment de la quantité de connaissances que l'hôte a sur le code. Il est aussi important de définir une évaluation des algorithmes de masquage.

8.10 Estimation de L'état

La technique d'estimation de l'état ‘’**State Appraisal**’’ est utilisée pour protéger une plate-forme par rapport à un agent dont l'état aurait été modifié par une plate-forme défaillante ou ennemie. A ce titre, elle pourrait aussi apparaître dans les techniques de protection de l'agent, car elle permet de détecter une altération de son état. L'auteur et l'utilisateur de l'agent seront protégés contre une utilisation frauduleuse de leur agent. Farmer, Guttman et Swarup [24] ont proposé l'idée de fonction d'estimation de l'état qui prend comme entrée l'état de l'agent et qui donne en sortie un ensemble de ressources dont l'agent a besoin pour exécuter sa

tâche. Le producteur de l'agent et l'utilisateur créent leurs propres fonctions. Chacune est signée numériquement pour empêcher toute modification puis est liée à l'agent. L'auteur écrit une fonction **max** qui donne le maximum de ressources que l'agent peut demander. L'utilisateur de l'agent écrit une fonction **req** qui donne les ressources que demande l'agent; cette fonction peut minimiser le coût que l'utilisateur devra payer pour les ressources de son agent. S'il n'y a pas de modifications de l'état de l'agent, on aura **req(R)** inclus dans **max(R)**. Si ce n'est pas le cas, c'est qu'il y a eu modification de l'état de l'agent mobile. Ces fonctions permettent aussi à la plate-forme de savoir de quelles ressources l'agent a besoin et de les lui allouer si cela est conforme à sa politique de sécurité. Cependant, la manière dont cette théorie sera mise en pratique n'est pas claire car l'ensemble des états possibles d'un agent est très grand, et écrire des fonctions d'estimation de l'état sera certainement facile pour détecter les attaques évidentes; pour des attaques plus subtiles, cela devient plus compliqué. De plus, il n'est pas toujours possible de distinguer un résultat normal d'une tentative de tromperie.

8.11 Limitation du temps d'exécution

Esparsa [17] propose une méthode de limitation du temps d'exécution de l'agent mobile dans chaque hôte visité. Elle commence par établir un référentiel global de temps. Parce que dans la pratique il est impossible de faire synchroniser réellement les horloges des plates-formes, la méthode amorce l'activité d'un évaluateur du temps de transmission afin d'estimer le décalage entre le référentiel global de temps et le temps de référence de chaque plate-forme, et la durée de transmission nécessaire entre de hôtes successifs. Enfin, la méthode emploie un estimateur de calcul du temps nécessaire à l'exécution entière de l'agent mobile au niveau de chaque hôte. Un hôte est déclaré non malicieux si son temps d'exécution réel et son temps de transmission réel sont confinés respectivement dans la limite:

1. du temps d'exécution estimé.
2. de la durée qui sépare son arrivée sur le nouvel hôte de son départ du prédécesseur.

Cependant, la méthode reste inefficace pour deux raisons principales : le coût élevé de l'estimation du seuil de l'erreur sans écarter le risque de faire un mauvais choix, et la contrainte de maintenir une connexion avec l'agent et ce durant tout son itinéraire.

9. Synthèse

Au sujet de la protection de la plate-forme, beaucoup a été fait. En particulier, le code avec preuve constitue une idée intéressante qui peut être améliorée à deux niveaux : optimiser la méthode pour diminuer la taille de la preuve et trouver de nouvelles propriétés qu'elle peut prouver. Par ailleurs, l'estimation de l'état mériterait une mise en pratique car, si l'idée en soi est bonne, la mise en pratique réelle n'a pas été décrite jusqu'ici. La solution préconisée compose avec plusieurs techniques déjà citées. L'authentification du certificat combinée à un moniteur de référence, chargé de superviser et contrôler le niveau des accès, et des méthodologies, pour estimer le degré de confiance de l'agent mobile, et pour associer un acte avec son propriétaire sont autant des clés d'une solution durable et efficiente.

Cependant, les résultats obtenus concernant la protection de l'agent ne sont patentes que dans un domaine très restreint. Ainsi, la boite noire limitée dans le temps n'a donné lieu qu'à des algorithmes de confusion (Obfuscating Algorithms) très simples. Le calcul de fonctions cryptographiques est aussi possible mais uniquement pour les fonctions polynomiales et rationnelles. Dans ces deux cas**,** il serait intéressant de trouver des algorithmes applicables à un contexte plus large. Pour la boite noire limitée dans le temps, un moyen d'évaluer les algorithmes serait nécessaire. D'autres techniques de protection sont plus applicables. L'utilisation des traces cryptographiques est aussi une idée intéressante que Vigna [14] a appliquée et qui donne des résultats certains. Cependant, elle détecte un grand nombre d'attaques au prix d'une demande excessive en mémoire et d'une vérification coûteuse en temps à base d'une réexécution simulée de l'agent. Le risque de faire des extra vérifications systématiques inutiles par manque d'indications ne fait qu'augmenter son coût. Deux questions demeurent toutefois en suspens : comment protéger les entrées qui sont fournies par la plate-forme à l'agent et comment améliorer le protocole notamment pour éviter l'attaque de deux hôtes ennemis qui se suivent sur l'itinéraire de l'agent. Quant à la technique de limitation du temps d'exécution proposée par Esparza [17], elle reste inefficace pour deux raisons principales : le coût élevé de l'estimation du seuil de l'erreur sans écarter le risque de faire un mauvais choix, et la contrainte de maintenir une connexion avec l'agent et ce durant tout son itinéraire. L'inconvénient majeur du vote de Minsky est, bien sûr, les ressources additionnelles

qui sont consommées par la duplication. On peut rajouter à cela qu'un prédicat divergeant empêcherai l'envoie de résultats aux prochaines plates-formes.

CHAPITRE 2

PROTOCOLES RESILIENT AUX ERREURS & SYSTEMES OO-ACTION

1. Introduction :

Les techniques de prévention utilisent le concept de l'intégrité du traitement du code mobile afin d'en assurer la protection des agents mobiles. Avant, le travail de Thomas Sander [32], les chercheurs croyaient que l'implantation de l'intégrité du traitement de ce code n'est possible qu'à travers un matériel de confiance. Bien que, la solution de Sander se limite uniquement aux fonctions polynomiales et rationnelles, ce travail a permis de consolider l'idée de la mise en œuvre logicielle de l'intégrité. Par la suite, plusieurs techniques logicielles ont vu le jour, variant des techniques d'obscuration et masquage jusqu'à la technique de résistance aux erreurs.

L'obscuration du code mobile décrit une méthode de type boîte noire pour protéger un agent contre une plate-forme ennemie. Elle consiste à modifier le code de l'agent de telle sorte qu'il fasse le même travail, mais que la lecture du code ne permette pas de comprendre ce que fait la fonction ou de la modifier sans détection. Le problème est qu'il n'existe pas de solution complète d'obscuration. Le calcul de fonctions cryptographiques est une solution partielle qui ne peut pas être appliquée à tous les agents. Ainsi, Hohl [33] introduit le concept de boîte noire limitée dans le temps. Les propriétés sont les mêmes que celles de la boîte noire, mais elles ne sont valables que pour un certain temps qui est lié à l'agent. Dans ce cas, il faut tout d'abord que le code de l'agent change à chaque début de l'intervalle de protection.

La technique de masquage du code et la résistance aux modifications est définie par Bazzi [38]. Elle consiste à rajouter du code de manière systématique et automatique en gardant la même fonction. Une solution pour les fonctions polynomiales est proposée dont le masquage et la résistance aux modifications sont prouvés. Le problème du masquage du code pourrait être mieux spécifié en tenant compte notamment de la quantité de connaissances que pourrait avoir l'hôte sur le code. Il est aussi important de définir une évaluation des algorithmes de masquage.

L'obscuration et le masquage sont des boites noires pour protéger un agent mobile là où le code y est confiné pendant son exécution. Par conséquent, la capacité de traçabilité des

actes malicieux est quasiment impossible. La traçabilité de ces actes est une condition sin quinone au procédé de médiation afin d'assainir Internet de ces hôtes dits malicieux.

Contrairement aux techniques d'obscuration et de masquage, la technique de résistance aux erreurs a innové dans de le domaine de prévention d'actes malicieux des hôtes. L'idée principale de cette technique est de masquer les effets des actes malicieux. La confidentialité du code mobile est garantie non pas par la notion de la boite noire mais plutôt en tolérant les erreurs dues aux actes malveillants des hôtes (i.e., résilient aux erreurs). De plus, de par sa démarche, cette technique présente un vrai potentiel de traçabilité des actes malicieux. Ce potentiel a été exploité pour développer un processus de médiation [42]. La médiation est considérée comme un évènement d'effet de bord car la fiabilité des exécutions des agents mobiles est garantie par la technique d'origine. L'objectif est d'instaurer un ordre et une régulation au niveau d'Internet. La régulation et l'ordre sont en relation directe avec la gestion des réputations des hôtes résolument déterminés malicieux. En réalité ces hôtes ne sont que des hôtes qui se comportent en dehors d'un consensus commun. En règle générale, ce dernier mérite une plus grande confiance. Il est considéré comme la norme naturelle des exécutions sans erreurs, i.e. la norme des hôtes non malicieux.

Ce chapitre a pour but d'introduire la technique du vote et de la duplication qui résiste aux erreurs et d'explorer le cadre formel sous-jacent à l'adjonction du processus de médiation.

2. Vote et Duplication

Le paradigme des agents mobiles est proposé pour une variété d'applications au niveau d'Internet et les systèmes distribués de grande échelle. Cependant, les facilités liées à ce sujet sont peu abordées à un niveau systèmes d'exploitation. L'un de ces aspects est la réalisation de la résistance aux erreurs sans un matériel spécialisé.

Dans un système distribué ouvert, les agents d'une application doivent non seulement survivre les erreurs (ou actes malicieux) des hôtes visités, mais aussi être résilient à des actes hostiles venant d'autres hôtes. La fiabilité des traitements ne doit dépendre que des hôtes sans erreurs visités. On suppose que les hôtes hostiles peuvent produire des messages erronés, prétendre êtres d'autres hôtes ennemis, mais qu'ils n'ont jamais accès aux secrets d'hôtes sains.

La duplication et le vote sont nécessaires pour neutraliser les actes malicieux des hôtes visités. Des protocoles de la duplication et le vote sont explorés dans une variété d'applications. Les expériences menées révèlent un fait, celui que les hôtes sains et rapides masquent les délais imposés par ceux qui sont plus lents. Par conséquent, la duplication améliore les performances de quelques applications.

2.1 Pipeline Résilient aux Erreurs

Un simple scenario d'un calcul à base d'un agent mobile consiste en une succession de visites de cet agent à une liste de hôtes, qui reviendra probablement par la suite à sa base d'origine pour délivrer ses résultats. Les difficultés soulevées pendant l'opération de rendre ce simple scenario résistant aux erreurs sont typiques à ceux associées à des calculs à base d'agents beaucoup plus complexes. Par conséquent, ce type de traitement simple est retenu comme point de départ à cette exploration.

Le scenario de calcul ci-dessus est assimilé à une exécution du style ''*Pipeline''*. Dans la figure suivante les nœuds représentent les hôtes, les arcs décrivent le mouvement d'un agent d'un hôte à un autre. Chaque nœud correspond à une phase du *Pipeline*. S est la *source* du *Pipeline* ; A est la *destination* qui généralement se confond à la source.

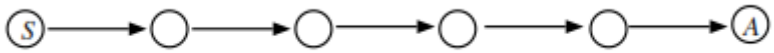

Figure 1: A simple agent computation

Ce Pipeline de traitement d'un agent n'est pas résilient aux erreurs. La fiabilité d'une phase dans ce Pipeline dépend de la fiabilité de la phase qui la précède. Un simple acte malicieux peut se propager jusqu'à la destination.

Une première étape vers l'accomplissement de la résilience aux erreurs est la duplication de chaque phase. On suppose que l'exécution à chaque phase est déterministe, sans autant, être capable de connaitre à l'avance les composants de chaque phase. Ces derniers dépendent des résultats calculés dans les phases précédentes. L'entrée d'un nœud *p*

de la phase *i* qualifie les entrées majoritaires des nœuds de la phase *i-1*. De même, un nœud *p* de la phase *i* injecte sa sortie à chaque nœud de la phase *i+1*. Cette exécution qui est résiliente aux erreurs est illustrée dans la figure suivante :

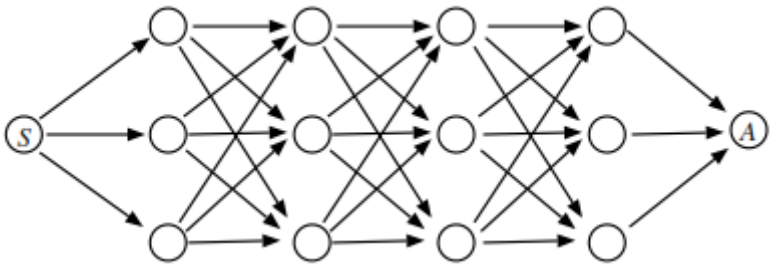

Figure 2: Replicated agent computation with voting

Il faut noter que le calcul dans la figure précédente masque plusieurs erreurs que de celle d'une architecture dont le Vote n'est opéré que juste dans la phase qui précède l'arrivée au nœud de destination. Le vote à chaque phase, limitera l'impact des dégâts occasionnés au traitement de l'agent dans une phase de l'exécution, dans les exécutions des phases suivantes. Plus précisément, l'architecture de la figure précédente masque les valeurs erronées des clones minoritaires de l'agent et ce à chaque phase.

2.2 Performances du Vote

Le Vote engendre des coûts additionnels de performances. En premier lieu, il y a un coût intrinsèque associé à l'envoi, la réception, et la comparaison des Votes. En deuxième, il y a lieu d'inclure le délai de synchronisation. Un agent ne vote i.e., émettre une sortie qu'après la réception de valeurs identiques majoritaires. Cependant, le Vote améliore quelques fois les performances car avec le vote le traitement n'est pas obligé d'attendre l'agent fiable le plus lent ; en effet, un agent qui Vote n'a besoin d'attendre que l'arrivée d'un nombre majoritaire de valeurs identiques. Par conséquent, c'est l'agent médian, plutôt que, l'agent le plus lent qui détermine la rapidité d'exécution du traitement.

Dans ce sens, la première expérience menée par Minsky[16] explore le coût du Vote dans un système uniforme. L'intérêt est porté sur l'influence d'une faible fréquence de Vote sur l'amortissement du délai de synchronisation. Ainsi, Les agents procèderont au Vote après une séquence de visites aux hôtes et non pas à chaque fin d'une phase. Les résultats de l'expérience obtenus sont représentés par un graphe sous la forme d'un Histogramme donné ci-après :

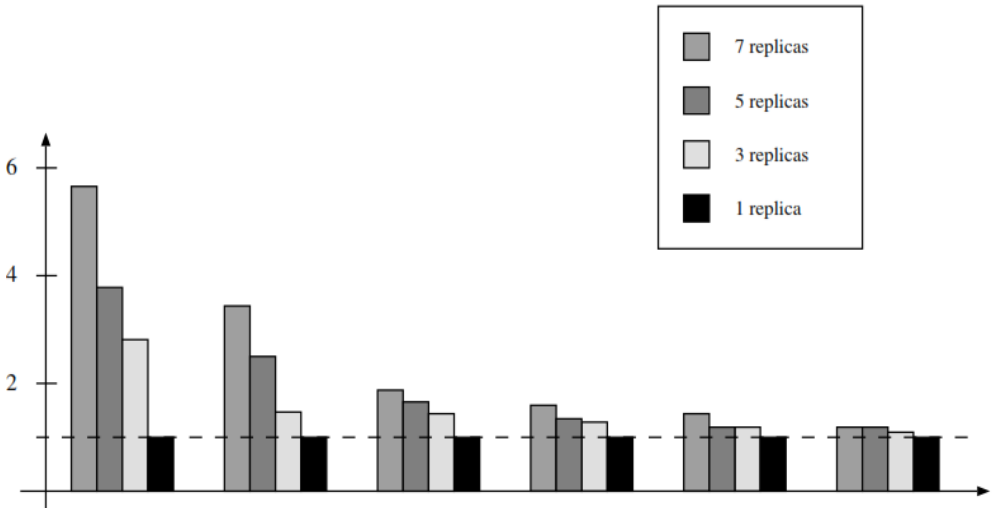

Figure 3: Voting performance with various voting frequencies

Le graphe montre des améliorations significatives, lorsque le groupe de la séquence des hôtes visités augmente d'une composante à un hôte jusqu'à dix. Au-delà, de ce nombre l'amélioration et plutôt non significative.

Lorsque le Vote est inhabituel, on peut craindre que le coût du Vote ne devienne élevé à cause d'un plus grand délai de synchronisation. Quand le vote est rare, les temps nécessaires à un agent de voter se singularisent progressivement appart, dû au délai de synchronisation qui augmente. Cependant, dès qu'un agent Vote, il n'attend qu'une simple majorité et donc le Vote se termine au moment où le clone médian votera. Par conséquent, le temps nécessaire de l'achèvement d'un traitement d'un clone, dans un contexte de vote rarissime, est approximativement identique à celui où il n'y a qu'un clone à voter. Ce comportement est

confirmé par l'histogramme de la figure précédente. Mieux encore, le Vote peut rendre le traitement d'un clone plus rapide. Ceci s'explique par le fait que le vote est résilient aux hôtes qui sont lents. La figure suivante concerne les performances réalisées avec une probabilité de 2% de rencontrer un hôte lent et un Vote après chaque cinq déplacement, le facteur de lenteur est de 350 pourcent. L'histogramme suivant qui représente les résultats de la simulation conduite, montre clairement qu'aux delà de trois clones le ralentissement des traitements est nettement réduit et qu'il approche la valeur zéro avec sept clones.

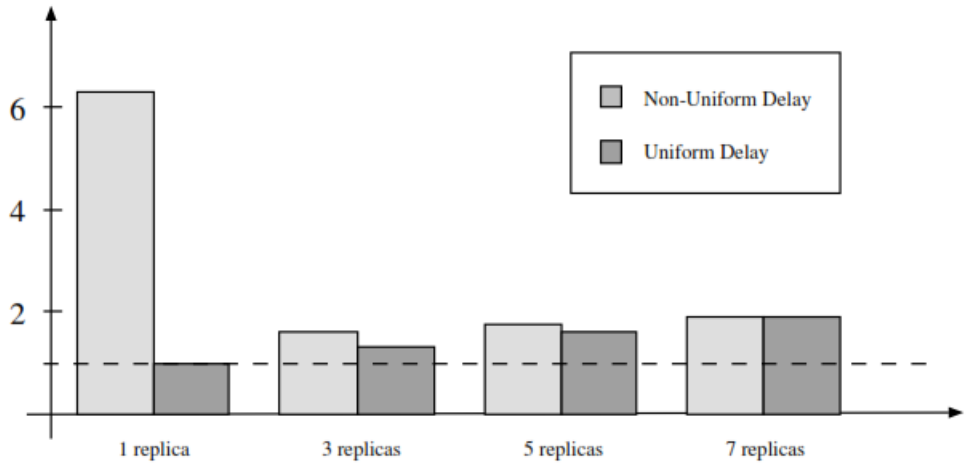

Figure 4: Voting performance with uniform and non-uniform delays

Le protocole de vote qu'on vient de décrire ne garantit pas l'intégrité des identités des agents du vote. Par conséquent, un procédé d'authentification distribué est nécessaire, plus particulièrement, la technique du secret partagé est utilisée. Elle facilite l'authentification des agents mobiles i.e., limiter le vote aux seuls agents mobiles éligible**s.**

L'adjonction du procédé de médiation proposé, est développé sur des bases mathématiques dans un cadre formel qui est le '' OO-Action ''. La suite de ce chapitre examine en détails ce cadre.

3. Les Systèmes OO-Action

Plusieurs langages et modèles de coordination d'agents indépendants ont été proposés dans la littérature [56, 57]. Linda [58] et Manifold [59] sont des exemples de langages dédiés à la coordination. Notre choix est porté sur les langages qui sont utilisés non seulement pour décrire l'aspect coordination mais aussi celui du traitement des systèmes. La coordination est modélisée dans un cadre formel appelé ''OO-Action Systems''. Dans OO-Action la coordination et le traitement sont traités d'une manière uniforme sans le recour à de nouvelles constructions du langage. Ensuite, l'expansion d'OO-Action Systems s'y assignée le nouveau but de modéliser l'aspect mobilité. Cette extension est réalisée à base de nouvelles formes de classes en accordant un intérêt particulier à la notion de localisation et de mouvement. Les objets de ces classes sont appelés objets mobiles. Un objet mobile constitue la granularité de base de la mobilité. L'objet mobile a un état qu'il déplace avec lui lors de son mouvement d'un endroit à un autre. Les valeurs possibles des localisations définissent le domaine de mouvement d'un agent mobile. Il faut rappeler que le domaine de mouvement d'un agent mobile varie et peut être très grand. Par conséquent, il est divisé en des ensembles plus petits appelés cellules. Chaque cellule est administrée par un objet coordonnateur dérivée de la classe coordonnateur. Le coordonnateur est responsable de programmer les évènements et de superviser les objets mobiles de telle sorte à interdire les irrégularités dans la cellule du coordonnateur, et de manière transparente à l'objet mobile. Ainsi, les systèmes OO-Action sont étendus par la mobilité incluant une forme adéquate et spéciale d'un coordonnateur. Il est signalé que les systèmes OO-Action sont définis à l'aide d'une sémantique de transformation de prédicats [62] et de calcul de raffinement. Par conséquent, le raisonnement sur ces systèmes est outillé par la notion de raffinement du calcul de raffinement. Le calcul de raffinement a un fondement mathématique solide qui nous permet de dériver formellement:

1. Les classes de mobilité des classes ordinaires,
2. Les coordinateurs décentralisés de coordinateurs centralisés.

Les transformations effectuées sont vérifiées à l'intérieur du calcul de raffinement ou par des règles de transformation pré-vérifiées à l'avance.

3.1 Les systèmes OO-Action Distribués

Cette section présentera le formalisme des Systèmes OO-Action avec une attention particulière portée sur la distribution. Un système OO-Action consiste en un ensemble fini de

classes. Chaque classe est une spécification du comportement d'objets crées dynamiquement et dont l'évolution est parallèle. Premièrement, une présentation du langage de spécification des classes et des objets est développée. Ensuite, une description de l'adéquation de l'approche OO-Action à la distribution est donnée.

Le langage : On considère Attr un ensemble d'attributs (variables) avec un ensemble non vide de valeurs. Le langage d'instructions est défini par la grammaire

S::= abort | skip | x := v | p | if b then S_1 else S_2 fi | S_1 ; S_2

où x est une liste d'attributs, v une liste de valeurs (peut être le résultat de l'évaluation d'une liste d'expression), b un prédicat et p le nom d'une procédure. Intuitivement 'abort' est une instruction d'inter blocage, 'skip' est la ''stuttering statement'', 'x := v' est assignation multiple, 'p' est un appel de procédure, 'S_1 ; S_2' est une composition séquentielle des deux instructions 'S_1' et 'S_2' et 'if b then S_1 else S_2 fi' est l'instruction alternative. Une déclaration d'une procédure p = P consiste d'une en tête p et d'un corps P. La sémantique de ces instructions est définie formellement (well-defined) via le transformateur de la faible pré-condition *wp* [53, 54].

Les Classes et les Objets : Soit respectivement CName et OName un ensemble fixé de noms de classes et de noms d'objets. Une classe est une paire (c, C), où c Є CName est le nom de la classe et C son corps composé de collection de données (attributs et variables d'objets), services (procédures et méthodes) et un comportement (un ensemble d'actions exécutés par les objets instanciés de la classe) :

C = |[attr $x := x_0$

obj $n := n_0$

meth $m_1 = M_1 ; \ldots ; m_h = M_h$

proc $p_1 = P_1 ; \ldots ; p_k = P_k$

do A **od**]| (1)

Le corps de la classe consiste de l'action $A = \|\|_I A_i$ du choix non déterministe parmi les actions 'A_i' avec $i \in I$. La première déclaration concerne les attributs locaux de la liste x décrivant les variables locales à un objet instance de la classe ; ce qui veut dire qu'elles ne sont utilisées en dehors de l'instance. Ces variables sont initialisées aux valeurs de la liste x_0.

La liste n composée de variables d'objets décrit un type spécial de variables locales d'un objet instance de la classe. Elles contiennent des noms d'objets utilisés pour appeler des méthodes d'autres objets. Les variables d'objets sont initialisées aux valeurs de la liste n_0. Nous acceptons que x et n soient disjoints deux à deux.

Une méthode $m_i = M_i$ est une procédure d'un objet instance de la classe. Elle est appelée par des actions de l'objet ou à partir d'autres objets instanciés d'autres classes. Une méthode consiste d'une en tête 'm' et d'un corps 'M'. Ce dernier est une action de type

$A :: = b \rightarrow S \mid x : \in V \mid \|\|_I A_i$. L' action '$b \rightarrow S$'est une instruction gardée et exécutée lorsque b est égal à *true* i.e., armée. $x : \in V$ est une abréviation d'une assignation non déterministe.

Une procédure $p_i = P_i$ décrit une procédure locale à l'objet d'une classe. Elle ne peut être appelée que par les actions de l'objet lui-même. En accord avec les méthodes, une procédure consiste d'une en tête 'p' et d'instructions composant un corps 'P'.

L'action spécifiée dans la boucle **do** …. **od** est une spécification du comportement autonome qu'a un objet de la classe pendant la phase de l'exécution. En particulier, une invocation d'une de ces méthodes par un autre objet n'est permise que si cet objet a la référence correspondante du premier objet, i.e. sa variable d'objet.

Les systèmes OO-Action Un système OO-Action OO consiste d'un ensemble fini de classes

$$OO = \{(c_1, C_1),\ldots\ldots,(c_n, C_n)\} \qquad (2)$$

où les actions dans chaque C_i y compris ses méthodes et ses procédures ne contiennent pas des instructions *new*, se référant à des noms de classes inutilisées par les classes de OO.

Il y a quelques classes marquées par un astérisque *. L'exécution démarre par la création d'une instance d'objet de chacune d'elles. Chaque objet à un état local. Un objet créé exécute ses actions qui sont armées selon un choix non déterministe. Comme les actions sont atomiques, leurs exécutions à base d'attributs et de variables d'objets disjointes progressent en parallèle. L'interaction entre objets se fait par l'appel de méthodes. En réalité l'appel d'une méthode est régit par le principe de la substitution. Cela permettra aux deux objets appelant et appelé d'exécuter conjointement et de manière atomique l'action en question.

Les systèmes OO-Action Distribués Conceptuellement, un système OO-Action est une composition parallèle de plusieurs objets, où chacun représente une instance d'une classe. Non seulement le parallélisme est possible à l'intérieur d'un objet mais il l'est aussi parmi des objets différents. Initialement, les seuls objets générés à partir des classes spéciales avec astérisques sont actives. Ces actions peuvent activer d'autres objets en exécutant une instruction *new* correspondante. Ceci traduit l'exécution dans un système distribué à base d'objets.

3.2. Spécification de la mobilité

Dans le paragraphe précédent, des objets différents associés et exécutés par des nœuds d'un réseau de processus composent ce qu'on a appelé explicitement un système OO-Action distribué. Cette section introduit la notion d'objets mobiles et décrit comment on modélise un système dont les objets migrent entre les nœuds d'un réseau.

Objets Mobiles Les systèmes qu'on souhaite modéliser consistent en un nombre arbitraire d'agents communicant, se déplaçant de manière libre dans un domaine et accomplissent un traitement désiré. La différence majeure avec les systèmes distribués est que les objets ont la capacité de se déplacer et leurs comportements peuvent varier d'un endroit à un autre. La communication entre objets dépend entièrement des positions courantes. Autrement, ils se comportent comme de simples objets distribués. Ainsi, ils transportent leurs états locaux. Mieux encore, lorsqu'un objet arrive à sa nouvelle destination du domaine, il devient actif et cesse d'exister dans la source.

Soit **OLoc** le domaine de mobilité. Le langage des objets distribués est alors étendu avec deux nouvelles instructions :

S :: = … | move(OList, Lexp) | l := move(OList, Lexp) (3)

Lexp est une expression évaluant à une location dans le domaine OLoc et OList est une liste de noms d'objets incluse dans OName. L'instruction move(OList, Lexp) évalue l'expression Lexp et les objets de OList sont déplacés à la destination résultante. Par contre, l'instruction l := move(OList, Lexp) range en plus le résultat de l'évaluation dans l'attribut spécial l. Le prédicat

at(OList, Lexp)

est utilisé à des fin de vérifications de la domiciliation des objets de OList dans la location physique obtenue à partir de Lexp. Quand OList est absente, l'objet appelant est l'objet cible du mouvement ou de l'évaluation du prédicat. Lexp est une forme plus générale d'exprimer une localisation physique. Maintenant on est en mesure de donner la définition suivante :

Définition Une classe de mobilité (mc, MC), est une classe de la forme :

MC = |[**loc** l :R

attr x := x_0

obj n := n_0

meth $m_1 = M_1$;…… ; $m_h = M_h$

proc $p_1 = P_1$;…….. ; $p_k = P_k$

do A **od**

]| (4)

où l est un attribut local, A une action, les corps des méthodes, et les corps des procédures peuvent contenir la forme précédente des instructions du langage étendu (3). Un objet mobile est une instance de cette classe.

L'attribut de la localisation l décrit un type spécial de variable locale d'un objet mobile. Elle mémorise les valeurs de localisations représentant la position de l'objet. Le type R du domaine OLoc de l'attribut de localisation est obligatoire et il dénote le domaine des déplacements de l'agent mobile. Cet attribut peut être initialisé à une position initiale de l'objet.

Le domaine OLoc de la mobilité est une abstraction forte des valeurs possibles de la notion de localisation. Il peut ainsi modéliser des nœuds d'un réseau, des rues, des cartes ou les trajectoires d'un avion. L'élément de base à cette vision repose sur le fait que le type-engine de ce formalisme est du type ''logique d'ordre supérieur'' [54]. Par conséquent, le type de la localisation est un type général.

Les changements de valeurs de l'attribut de la localisation et aussi l'exécution d'une instruction move modélisent le mouvement des objets à d'autres positions de leur domaine. Le prédicat at(OList, l) est aussi utilisable soit avec des gardes, soit avec d'autres expressions booléennes afin de déterminer si la position actuel d'un objet est la localisation l. Ces gardes sont appelés les gardes de la localisation.

En réalité le langage est étendu d'une manière à ce que le concept des objets mobiles est embarqué dans le cadre formel des objets distribués. L'argumentation stipule que les éléments rajoutés ne sont qu'un nouveau type d'attributs locaux et les nouvelles instructions sont assimilés à des opérations d'assignations uniquement.

Les systèmes d'action orientés mobilité (MOAS)

Définition Un système d'action orienté mobilité est un ensemble fini de classes

$MO = \{ (c_i, C_i)_{i\in\{1,\ldots,n\}}, (mc_j, MC_j)_{j\in\{1,\ldots,s\}}\}$ (5) où chaque (c_i, C_i) est une classe au sens de la définition (1) et chaque (mc_j, MC_j) est une classe de mobilité (4).

Dans ce système, les objets mobiles ne communiquent que par le biais de l'invocation de méthodes avec d'autres objets, s'ils sont au niveau d'un certain voisinage (vicinity). Par conséquent, la visibilité réciproque de deux agents, dont au moins l'un des deux est mobile a besoin d'un contrôle via un garde de localisation. A moins que ce garde ne soit égal à vrai, les deux objets sont réciproquement invisibles bien qu'ils possèdent leurs références mutuelles via les variables d'objets. Lorsque le garde de la localisation devient vrai la communication via l'invocation de méthodes qui prendra place. Le domaine défini par le garde de la

localisation est perçu comme une notion de proximité de l'objet où la communication peut avoir lieu.

3.3. Coordinateurs

Les systèmes d'action orientés mobilité qu'on vient de décrire ont un modèle d'exécution non déterministe. Ceci est approprié à un niveau des spécifications. Les patterns de la coordination ont pour objectif d'introduire des restrictions afin de réguler ce non déterminisme. Pour cela, des classes coordinatrices sont spécifiées sans que la coordination introduite n'affecte la fonctionnalité des autres classes du système. La coordination est prise en charge par les classes du coordinateur à travers les appels de méthodes aux objets nécessiteux. La question principale est comment coordonner un ensemble dynamique d'objets. Dans le contexte de systèmes orientés mobilité, le besoin à des coordinateurs avec un champ d'actions réduit est ressenti.

3.4. Objet orienté et Coordination

Le modèle d'exécution sous-jacent les systèmes OO-Action est actuellement non déterministe. Par conséquent, la programmation à l'exécution de certaines actions n'est pas garantie. Cependant, lorsque on spécifie des coordonnateurs est que l'on souhaite forcer l'exécution des actions spécifiques du coordonnateur, le principe utilisé de la coordination est définie en termes d'implantation de priorités dans la composition des actions. Le mécanisme consiste en un opérateur puissant de composition d'actions, d'objets, et de classes, destiné aux systèmes orientés objet.

Coordination des actions Considérons deux actions A = a → S et B = b → T. L'établissement d'un ordre de priorité de leur composition notée A // B est définie comme

$$A \,//\, B \equiv (a \rightarrow S) \,||\, (\neg\, a \text{ and } b \rightarrow T) \qquad (6).$$

Nous dirons que A coordonne l'action B. En d'autres termes, l'action A a une plus grande priorité sur l'action B, ainsi A est exécutée lorsque elle armée, alors que B peut être exécutée quand A est désarmée.

La définition ci-dessus étend le langage d'actions par l'action suivante :

$$A ::= \ldots\ldots \mid A \,//\, A.$$

Coordination des objets Elle est établit maintenant au niveau objets. Pour qu'un objet o_1 influence un objet o_2, il faut que o_1 possède la référence de o_2 via une variable d'objet par exemple n. Ensuite o_1 peut appeler une méthode de o_2. Afin que o_1 influence o_2 et forcer l'exécution de A il faut bien évidemment que A coordonne les actions de o_2.

Définition Soient deux objets o_1, o_2 € OName. On dira que l'objet o_1 coordonne l'objet o_2, dénoté par o_1 // o_2, si o_1 a une référence à o_2 via une variable d'objet, et les actions de o_1 coordonnent les actions de o_2. o_1 est appelé objet coordonnateur et o_2 objet coordonné.

Coordination entre les classes Une classe (c_1, C_1) coordonne une autre classe (c_2, C_2), dénoté par (c_1, C_1) // (c_2, C_2), si chaque instance d'objet de c_1 est le coordonnateur de chaque instance d'objet de c_2. c_1 est appelé classe coordinatrice et c_2 classe coordonnée.

Lemme Soit (c_1, C_1) // (c_2, C_2). Alors un objet instancié de c_1 est un coordonnateur d'un objet instancié de o_2 si le premier détient une référence à l'objet instancié de c_2.

3.5. Coordination dans les environnements à mobilité

Considérons maintenant la coordination dans les classes à mobilité et comment coordonner les objets mobiles où l'ensemble des objets visibles est dynamique. Il est absurde que le coordonnateur visionne le tout, ici, le coordonnateur n'est au courant que des objets d'un certain voisinage.

La classe Coordonnatrice Une classe coordonnatrice où le plus important est la variante mobilité est formulée ci-après :

Définition Soit (mc, Mc) une classe à mobilité. Soit R le type de l'attribut de la localisation dans MC et soit

$R = R_1$ union ……R_k des partitions de R. Une classe coordonnatrice (cc, CC) de (mc, MC) est une classe de la forme suivante :

CC = |[**cell** c : R

attr x : x_0

obj n : n_0

meth SetR; $m_1 = M_1$;……..; $m_h = M_h$

proc $p_1 = P_1;\ldots\ldots\ldots; p_k = P_k$

do A **od** **]|** (7).

où c est un attribut local constant utilisé dans les actions A, les corps des méthodes et des procédures. Ces derniers contiennent des instructions du langage étendu de la classe précédente. La classe a exactement k instances, appelées objets coordonnateurs, et le type de l'attribut **cell** du j^{ieme} objet instance est R_j, $j \in \{1,\ldots\ldots,k\}$ affecté par la méthode SetR(R_j) au moment de l'instanciation.

L'attribut **cell** est interprété comme étant une constante locale décrivant le domaine de visibilité du coordonnateur. Son type est le type de l'attribut de la localisation de la classe à mobilité qu'elle coordonne. Cependant, chaque objet coordonnateur a un élément de la partition comme type de l'attribut **cell** hérité implicitement par le mécanisme de sous typage. L'attribut **cell** reste inchangé tout le long de son cycle de vie, et il est initialisé lors de la création par la partition correspondante à l'aide d'un appel à la méthode SetR( ). La liste n contient des références aux objets coordonnés ainsi que d'autres références nécessaires. Ceci est toujours intégré dans le cadre formel du départ.

Les systèmes d'actions coordonnées et à mobilité (MoCAS) Notons que chaque classe coordonnatrice coordonne une seule classe à mobilité. Donc, nous avons la définition suivante :

Définition Un système d'actions coordonnées et à mobilité consiste en un ensemble fini de classes

$$MCO = \{ (c_i, C_i)_{i\in\{1,\ldots..,n\}}, (mc_j, Mc_j)_{j\in\{1,\ldots..,m\}}, (cc_j, CC_j)_{j\in\{1,\ldots\ldots,s\}}\}$$

Où $s \leq m$ et $(cc_j, CC_j)) // (mc_j, Mc_j)$, quel que soit $j \in \{1,\ldots..,s\}$, (c_i, C_i) est une classe au sens de la définition (1), (mc_j, MCj) une classe à mobilité (4), et (cc_j, CC_j) une classe coordinatrice (7). Chaque classe coordinatrice coordonne une certaine classe à mobilité dénoté par la clause where-clause. D'habitude MCO est dénoté par

$$\{(c_i, C_i)_{i\in\{1,\ldots.,n\}}, (mc_i, MC_i)_{i\in\{s+1,\ldots..,m\}}\} \text{ union } \{(cc_j, CC_j) // (mc_j, MC_j)\}_{j\in\{1,\ldots..,s\}}$$

MoCAS est un système d'action plus déterministe que ceux définis en (2) et (5) car les actions du coordonnateur s'exécutent de manière prioritaire que les actions de la classe respective à mobilité.

Le modèle qu'on vient de décrire est un modèle naturel et intuitif d'un système dynamique à objets mobiles. L'aspect dynamique est le résultat du fait que les coordonnateurs ont un contrôle sur les actions des objets mobiles à proximité, .i.e. ils détiennent leurs références. Hors, cet ensemble de références du coordonnateur aux objets mobiles varient d'un déplacement à un autre de ces derniers. Par conséquent, l'ensemble des objets référencés varient dynamiquement puisque les objets mobiles sont en perpétuelles mouvements.

4. Conclusion

Ce chapitre a retracé l'émergence des solutions préventives purement logicielles. Plusieurs solutions mentionnées sont sans intérêts particuliers à notre étude. Ce chapitre a décrit en détails la méthode de notre choix qui est celle de résistance aux erreurs et a développé le cadre formel du travail de la nouvelle activité de médiation proposée. Notre choix est influencé par un fait majeur celui d'observer que cette méthode de résistance aux erreurs est porteuse d'un précieux potentiel intrinsèque de traçabilité d'actes malveillants des hôtes. Cette possibilité de traçabilité est un élément clé à la désignation du processus d'effet de bord de médiation. Le chapitre suivant décrit les outils et les mécanismes par lesquels l'activité de la médiation est mise en œuvre dans la technique choisie**.**

Chapitre 3

Protocole Résilient aux Erreurs Etendu

1. Introduction

La technique de duplication et du vote [15, 16], résiliente aux erreurs a été retenue afin de promouvoir l'intérêt de l'activité de médiation. Cette technique appartient à la famille des techniques de prévention. Son choix délibéré repose sur son aptitude singulière et importante de cohabitation facile avec un procédé d'identification des hôtes défectueux. L'aptitude d'observabilité des évènements pendant les exécutions des agents mobiles est un point décisif. L'inaptitude d'observer ces évènements est une caractéristique qui domine le reste des techniques préventives. Les deux principes de bases de ces techniques sont soit : (1) le confinement de l'exécution ou (2) le masquage du code.

Dans la technique de duplication et du vote, les hôtes dont l'information est divergente lors d'un vote sont considérés malicieux. Nous considérons qu'une telle qualification distinctive est simple, directe et non flexible. Ce chapitre assigne le rôle de proclamer le verdict afin de qualifier un hôte par le caractère malicieux à un processus plus élaboré. Ce processus qui sera proposé par la suite se prétend d'être flexible, attentif et plus prudent. Dans ce processus, nous procédons à : (1) explorer la manière d'associer une qualité de confiance ou de suspicion à un hôte. L'attribution de la dite qualité doit être temporelle. (2) Exécuter le vote parmi les agents dont les informations collectées proviendront d'hôtes de confiance. L'information dont la source est un hôte suspect est écartée. (3) Déléguer la capacité de proclamer le caractère malicieux d'un hôte à un arbitre. La désignation d'un tel caractère requiert un dialogue et peut être paramétrée par un seuil de tolérance, .i.e. une médiation. (4) Capturer les comportements ''étranges'' des hôtes de confiance puis le disséminer dans le net. Le processus proposé est intégré dans la technique de Minsky[16] qui identifié l'occurrence des actes malicieux. En dépit de l'importance de la notion de confiance qui continue toujours de jouer un rôle primordial dans le développement de systèmes sécurisés, le sens associé au mot confiance d'un ''principal'' est rarement clairement défini. Un point central dans notre processus est le raisonnement technique, saisissant la confiance comme les preuves de coopération. Un propriétaire d'un agent mobile estime la confiance d'un hôte à base de son comportement correct, i.e., consistant à sa politique déjà publiée. Dans notre approche nous remplacerons le problème difficile de la détection des hôtes malicieux par un simple

processus de vérification de l'opportunité des explications de l'hôte. Nous avons l'ultime conviction que, soumettre les hôtes à une charge plus grande, a réduire de manière significative la tentation d'altérer l'agent mobile.

Pour atteindre notre objectif, nous allons enrichir la cadre formel de Mobile-OO-Action. Notre extension fournira alors deux actions, une action de clonage et une autre pour déplacer les clones chez des hôtes différents. La propriété d'atomicité est la caractéristique principale de ces deux actions. Ceci est de plus important car elle est:

1. Essentielle lors de la vérification quand les agents retournent à la base.
2. A la base de construction des spécifications expressives et non ambiguës.

Nous rappelons que le cadre formel de Mobile-OO-Action est le calcul de raffinement. Ainsi, nous pouvons dériver de manière formelle les implémentations des systèmes de leurs spécifications abstraites de haut niveau. Une fois les deux actions précédentes sont établies, nous développons un protocole rendant les agents mobiles altérés inéligibles au vote. Ce même protocole est de plus capable de déterminer les hôtes malicieux. A la base du protocole proposé est le protocole de duplication et du vote dont l'idée est de produire un traitement parallèle redondant. L'estimation de la confiance à l'aide de traitements parallèles redondants fait apparaitre l'agent mobile à son propriétaire comme une boite noire ''black-box agent''. Ce qui l'emporte à l'égard d'un propriétaire d'un agent mobile est toujours la fiabilité de l'information retournée par son agent. Par conséquent, une protection effective est apportée aux propriétaires des agents mobiles sans que la flexibilité des systèmes à agents mobiles ne soit affectée ou ne requiert un coût supplémentaire.

Ce chapitre est subdivisé en quatre sections. La première étant l'introduction, la deuxième section décrit sommairement les systèmes Mobile OO-Action. Dans la troisième section, l'extension apportée à Mobile OO_Action est donnée avec sa sémantique. La section quatre reprend quelques approches de la sécurité des agents mobiles et puis développe en détail notre proposition. Et nous terminons le chapitre par une conclusion.

2. Les Systèmes à Base du Système Mobile-OO-Action

Le système Mobile-OO-Action modélise un système avec un nombre arbitraire d'agents mobiles, pouvant se déplacer librement vers un site d'un domaine de mobilité pour ensuite exécuter un traitement particulier à volonté. Lugia[45] a étendu le formalisme des systèmes OO-Action avec celui d'une action de mouvement ''move'', un prédicat de positionnement, et une classe de coordination. La syntaxe de l'opération move est donnée ci-après :

S := move(Olist, L_{exp}) (1). l := move(Olist, L_{exp}) (2). où:

1. L_{exp} est une expression s'évaluant à une place du domaine de la mobilité Oloc, vers laquelle un déplacement est possible.
2. Olist dénote la liste des noms des objets à déplacer.

La première instruction évalue l'expression L_{exp} et déplace les objets de la liste Olist vers l'emplacement qui en résulte. L_{exp} est une expression donnant comme résultat un emplacement du domaine de mobilité Oloc. Quant à la deuxième instruction, elle range la valeur de l'évaluation dans un attribut spécial l. Dans l'extension, le prédicat at(Olist, L_{exp}) est aussi présent. Le prédicat est utilisé afin de vérifier la présence des objets de la liste Olist à l'emplacement de l'évaluation de l'expression L_{exp}. Lorsqu'Olist est absente, l'objet appelant est la cible de l'évaluation du prédicat. L'expression L_{exp} est une manière très générale de spécifier un emplacement.

Cependant, utiliser une telle opération de move de ''Lugia'' afin d'implémenter le protocole basé sur le principe de traitement redondant est incommode. Dans ce protocole, il y a nécessité, que le processus de déplacement des agents mobiles à des endroits différents possède la propriété d'atomicité ; sachant que, la phase finale du protocole est un processus de vote de majorité.

Lorsqu'on utilise l'opération de Lugia du mouvement, déplacer un nombre d'agents mobiles à des emplacements différents requiert une boucle d'itérations, où une instance de move prend un emplacement simple différent à chaque évaluation de l'expression L_{exp}. Le problème majeur avec ce schéma de fonctionnement consiste en la manière de gérer un échec d'une opération de move. Cela est d'autant plus important que, un simple échec d'une seule opération de ''move'' biaisera la tâche du protocole de vérification des agents. Par conséquent, le processus de vote de majorité est abandonné ou un quorum plus faible est

retenu. Les deux cas nécessitent un processus de nettoyage qui engendre une perte d'efforts et de temps.

Dans notre approche, nous proposons une opération de move unique. Cette action déplace plusieurs agents mobiles à plusieurs emplacements physiques. La nouvelle version de l'action de move est plus générale. Elle peut prendre en charge facilement celle de Lugia. Un deuxième avantage réside dans le fait que, caractériser cette opération par une propriété de tout ou rien est encore facile. Le code groupé qui en résulte élimine le besoin de considérer plutôt les actions des étapes individuellement. Par analogie et en appliquant le même raisonnement l'opération de création de plusieurs clones est dotée de la propriété d'atomicité.

3. Les Systèmes à Objets Mobiles Enrichis

Notre perception de l'action de ''move'' est différente de celle de Lugia. Toutefois, nous nous basons toujours sur le même ''background'' et la même syntaxe. Nous gardons, les concepts dont on a besoin pour modéliser la mobilité à l'intérieur du même formalisme. Le concept de la mobilité est encore un concept abstrait, qui peut être personnalisé par le concepteur à plusieurs environnements.

3.1 Principe

L'idée est de fournir une variété d'actions de move sous la forme d'une librairie de taille très modeste et qui est implémentée par une classe abstraite via un langage supportant la diffusion. Une action de cette variété est sous une forme qui s'adapte bien à la présence de Parre Feu, .i.e. Fire Walls, une autre pour accommoder la présence de SandBox, et une troisième afin de contrôler la portée des interactions. Notre intérêt est porté sur l'action de move dans un réseau qui incorpore des Parre Feux. La forme de l'action de move est plus générale, c'est-à-dire, d'être capable de déplacer plusieurs agents à des emplacements physiques différents. Le formalisme des systèmes Mobile-OO-Action est ainsi enrichi par l'instruction de move suivante :

Move(Olist, L_{exp}list).

En effet, cette action est puissante, elle englobe aussi la sémantique de l'ancienne action de move des systèmes à base d'objets mobiles. Il suffit dans ce cas, de composer juste la liste des L_{exp} d'une seule et unique expression. La nouvelle version de move utilise comme dans le cas

traditionnel le domaine de mobilité Oloc. Chaque élément de la liste des expressions est aussi une expression qui s'évalue à un emplacement dans le domaine Oloc. Finalement, Olist est inchangée et dénote toujours les noms des objets.

L'action de ''move'' qu'on vient juste d'introduire évalue la liste des expressions L_{exp}list et déplace aux destinations les objets de la liste Olist. Le déplacement des objets aux emplacements spécifiés est régi par une politique de correspondance de un à un .i.e. une correspondance bijective. Cette correspondance a comme domaine les valeurs de l'évaluation de la liste des expressions et comme Co-domaine les identités les objets, .i.e. agents mobiles de la liste Olist. L_{exp} est toujours comme dans les systèmes à objets mobiles, une forme très générale pour spécifier un emplacement.

D'autres aspects similaires sont soulevés par l'opération de créer des clones. Cette opération de création des copies conformes d'un agent doit aussi être atomique. Nous avons opté pour une démarche d'inscrire cette action comme une capacité intégrée pour plus de sûreté. Une des raisons est d'interdire toute mauvaise utilisation de duplication d'un agent. Cette capacité de duplication d'un agent peut être contrôlée, .i.e. supervisée par une action de coordination. Nous rappelons que cette capacité peut être activée et désactivée à bonne volonté par les agents. Cette nouvelle action de création de copies conformes des agents est consignée dans le même formalisme des systèmes Mobile-OO-Action. Ainsi les systèmes à objets mobiles sont enrichis par l'instruction suivante :

Clone(NClone).

Le processus de création de clones doit choisir un nouveau nom pour chaque clone. Les copies crées de l'agent dénotent une image figée de la structure de comportement. Cependant, il faut être prudent et ne pas considérer juste des alias, où les éléments de OName contiennent des références à des objets plutôt qu'à la valeur de l'objet [64]. Cela permet à deux ou plusieurs noms de dénoter la même référence et ainsi qualifier le même objet. Le processus de création des clones doit attacher des noms différents d'OName à des valeurs différentes de l'image figée de la structure de comportement. En ce qui concerne la supervision de l'action de clonage, l'agent initiateur de l'opération peut la contrôler de manière individuelle par une action de coordination. L'action de coordination active est désactive le clonage à volonté et à la volé. Pour clarifier plus l'idée, les deux structures de l'agent et du clone sont données ci-après :

Agent = Loc L :R

Attr x :: x0 Union Nclone

Obj n ≔ n0 Union clone-id, for i ∈ [1.. Nclone]

Meth m1 = M1 ….. mh = Mh

Proc p1 = P1 ……. Ph = Ph Union Clone(Nclone : integer)

Do

|| cloneid[i].setCanClone(False) /* Line A :1*/

|| od

Agent Specification

Clone = Loc L :R

Attr x ::= x0 Union Nclone Union canClone

Obj n ≔ n0

Meth m1 = M1 ….. mh = Mh Union SetCanClone(val can : boolean)(canclone = can)

Proc p1 = P1 ……. ph = Ph Union Clone(Nclone : integer)

|| canclone → Clone(Nclone) /* (line C : 1)

|| od

Clone Specification

On peut remarquer facilement que, malgré que la capacité de création de clones de l'agent clone est désactivée (Ligne A :1 & Ligne C :1) ; le traitement de l'agent clone n'est affecté en aucune manière. Ici, l'agent est dans un rôle de coordinateur, activant et désactivant l'action de clonage de l'agent clone dans des situations différentes à volonté et à la volée. La section suivante définit la sémantique des instructions rajoutées. Afin de garder le même cadre formel de travail, .i.e. le Mobile-OO-Action, les instructions seront définies selon les transformations du plus faible pré condition.

3.2 La sémantique de L'Extension

Deux aspects distinguent la nouvelle version de l'instruction '' move'' :

1. La propriété de l'atomicité.
2. Le paramètre L_{exp}list.

La propriété de l'atomicité est partagée aussi bien par l'instruction ''move'' que par l'instruction de clonage. Puisque, la propriété de l'atomicité est une notion déjà présente dans le ''move'' fourni par Lugia. Par Conséquent, seule la sémantique de l'opération de clonage sera définie. Ainsi, En traitant l'action de clonage nous sommes en train de résoudre implicitement et de manière partielle la nouvelle action de move. Le seul paramètre qui restera en suspend est le paramètre L_{exp}list. Ce paramètre par contre ne nécessite absolument aucun traitement spécial.

L'opération de clonage doit attacher comme nous l'avons déjà mentionné auparavant des noms d'objets différents à des valeurs d'objets différentes, dénotant la même image figée de la structure de comportement. Le mot différent signifie l'existence d'une propriété de ''jamais rencontré antérieurement'' où la structure de comportement est une ''fresh'' store et ''fresh'' stack. Définissons maintenant et de manière formelle la propriété de ''freshness'' :

fresh (ϕ , E) = error

fresh (e , E) = e if (e) is not in E

fresh (e U E_1 , E) = e if (e) is not in E

else fresh (E_1, E)

L'instruction de clonage est vue comme une transformation de prédicats, .i.e. une fonction reproduisant une pré-condition d'une post condition. Soient StackVar, StoreVar des ensembles dénombrables de ces structures particulières. Les deux ensembles StackVar et StoreVar sont des ensembles totalement ordonnés, donnant naissance à une correspondance un à un entre un ensemble de couple des variables (StackVar, StoreVar) et sa séquence ordonnée. Un procédé de nommage est alors une correspondance ''mapping'' de l'ensemble des paires (StackVar, StoreVar) aux noms d'objets d'Onames.

Au moment de la création des clones, une structure de comportement ''fresh'' est sélectionnée, .i.e. une paire de (StackVar, StoreVar) et puis l'attacher à un nom ''fresh'' du domaine Oname. Ensuite, l'image mémoire de l'agent appelant est copiée. Cependant, lorsque plusieurs clones sont à créer, aucun clone n'est créé quand au moins la création d'un seul clone est impossible. L'objectif est de préserver la propriété de l'atomicité.

Dans le treillis ''la lattice'' des programmes, un terme ''sequent'' d'une instruction d'une valeur ''magic'' est évalué à ''true'', ressemble à un effet équivalent où l'instruction n'a jamais été exécutée. Afin de rendre notre instruction atomique, la sélection de toutes les structures de comportements ''fresh'' prend la forme d'une coercition de la conjonction des sélections individuelles de tous les clones. La sélection s'évalue bien sûr à ''magic'' lorsque une seule sélection individuelle se solde par un échec ou un skip. La valeur ''magic'' désactivera l'action ultérieure. Cependant, en cas de succès, une séquence complète de substitutions de l'image mémoire de l'agent dans les structures de comportements ''fresh'' est réalisée. Enfin, une correspondance entre les structures de comportements ''fresh'' et les noms des objets est établie, dont une description concrète du processus est aussi ainsi établie :

Coercion($fresh_1$(StackVar, StoreVar)ᵔ…….ᵔ) ; Naming(([StackVar, StoreVar)$_1$\AgentCoreImage]),…….,)

Le symbole ‘’ ;’’ représente un opérateur de composition séquentielle, alors que le symbole ‘’\’’ qualifie un opérateur de substitution.

L’extension proposée est définie comme des transformations de prédicat. Par conséquent, elle adhère au même cadre formel du formalisme de Mobile-OO-Action .i.e. la transformation de la plus faible pré-condition. La section suivante détaille notre processus de détermination des agents mobiles altérés et les hôtes malicieux.

4. Protocole d’Exécution Parallèle

Avant de rentrer dans le vif du protocole d’exécution parallèle, nous jugeons utile de reprendre un peu de manière succincte le sujet de la sécurité des agents mobiles de manière générale.

4.1 Sécurité des Agents Mobiles

Le fait que l’information retournée par l’agent à son propriétaire ne peut être validée par ce même propriétaire, constitue l’obstacle majeur à une utilisation en masse des traitements basés agents mobiles [44]. Le travail de Sander concernant la sécurité des agents mobiles a remis en cause le postulat précédent. Il présente un protocole qui permet aux programmes de code mobile de s’exécuter crypté à l’exception de la partie composée d’instructions [32]. Par conséquent, l’exécution d’un agent mobile dans un système hôte est intègre et sûre. Cependant, ce protocole ne s’applique qu’aux fonctions rationnelles ou polynomiales. En pratique, il est impossible de prévenir les altérations des hôtes défectueux ou malicieux à l’encontre des agents mobiles transcrits en texte clair [44].

Nous, nous intéressons aux agents mobiles dont le texte est transcrit en clair pour des raisons de généralité. Notre approche définit la confiance à base de traitement parallèle redondant. Contre les attaques des hôtes, on utilise un protocole basé sur le principe de l’exécution parallèle. Avant de détailler le protocole, nous allons identifier en premier lieu quelques caractéristiques nécessaires de solutions plausibles qui maintiennent le potentiel et la flexibilité du paradigme de traitement à base d’agents mobiles.

L'approche la plus répandue de protection des agents mobiles et des hôtes consiste d'interdire simplement la migration vers des hôtes de non confiance et de ne pas accepter le code inconnu. L'approche est connue sous le nom de l'approche ''Trust''. Les techniques dont l'aptitude est de rendre les manipulations frauduleuses des agents impossibles ou plus difficiles tel ''Time Limited Black Box'' sont moins viables que celles dont l'effort cherche à déterminer l'occurrence des actes malicieux des hôtes [11]. Nous adhérons à la thèse que la survivance du propriétaire d'un agent est fortement compromise lorsque l'information retournée par son agent est incorrecte ou elle est arrivée en retard. Ceci est vrai car si le propriétaire de l'agent est induit en erreur de manière délibéré de par les informations de son agent, le propriétaire est certainement sous un assaut d'information. Lorsque cet assaut est contrarié, la survivance du propriétaire de l'agent est bien évidemment augmentée [11]. Vigna [14] a montré que ''tracking'' les états intermédiaires crées lors de l'exécution des agents mobiles est un ingrédient important pour produire des systèmes distribués sécurisés et flexibles. Les traces de Vigna peuvent être utilisées afin de déterminer les altérations possibles d'un agent mobile. Tandis que, Kassab [11] utilise la notion d'observabilité pour définir la notion de confiance. Une meilleure observabilité est accomplie par l'insertion des assertions protectives dans le code de l'agent mobile.

4.2 L'approche Proposée

Toute l'information d'un agent mobile est complètement disponible à l'hôte sur lequel il s'exécute. Alors, il est difficile d'appliquer les techniques de cryptographie traditionnelles pour détecter les altérations de l'agent. Du moment qu'un agent mobile ne peut pas protéger la clé cryptographique qu'il transporte à l'encontre d'éventuelles révélations durant son exécution chez un hôte ; l'agent ne peut pas utiliser des mécanismes cryptographiques pour vérifier la fiabilité de l'information rassemblée [44].

Empruntant le pas à Minsky [16], notre approche définie la notion de confiance ''Trust'' à base de traitement redondant. Un agent est vraisemblablement une boite noire pour son propriétaire. Ce qui est important d'un point de vue du propriétaire d'un agent est la fiabilité de l'information que cet agent retourne. L'information déroutante doit être écartée, l'intégrité de l'exécution ne doit dépendre que des hôtes intègres visités. Mettre à la disposition d'un propriétaire d'un agent mobile un mécanisme de vérification de la validité de l'information évitera évidemment de recourir à des mécanismes de traçabilité les états intermédiaires d'un

agent. Lorsque le propriétaire d'un agent est capable d'associer une information à son hôte source, il peut se plaindre lorsqu'il est induit en erreur.

Dans l'objectif de résoudre ce problème, Minsky [16] a proposé une approche qui se base sur la duplication et le vote. L'essentiel de cette approche est qu'elle considère les hôtes avec une information divergente au vote comme des hôtes malicieux. Nous considérons qu'une telle qualification distinctive est simple, directe et non flexible. Cette section assigne le rôle de proclamer le verdict de qualifier un hôte par le caractère malicieux à un processus plus élaboré. Ce processus proposé est flexible, prudent, et plus attentif.

Dans ce processus on tâchera : (1) explorer la manière d'associer une qualité de confiance ou de suspicion à un hôte. L'attribution de la qualité doit être temporelle. (2) Exécuter le vote parmi les agents dont les informations collectées proviendraient d'hôtes de confiances. L'information dont la source est un hôte suspect est écartée. (3) Déléguer la capacité de proclamer le caractère malicieux d'un hôte à un arbitre. La désignation d'un tel caractère requiert un dialogue et peut être paramétrée par un seuil de tolérance .i.e. une médiation. (4) Capturer les comportements ''étranges'' des hôtes de confiances et le répandre dans le net.

Le processus proposé est intégré dans la technique de Minsky[16] qui identifiè l'occurrence des actes malicieux. En dépit que, l'importance de la notion de confiance continue toujours à jouer un rôle primordial dans le développement de systèmes sécurisés, le sens associé au mot confiance ou un principal de confiance est rarement clairement défini. Un point central dans notre processus est le raisonnement technique, saisissant la confiance comme les preuves de coopération. Un propriétaire d'un agent mobile estime la confiance d'un hôte à base de son comportement correcte en concordance avec sa politique publiée. Un point crucial qui a été tout le temps négligé basé sur la détection des attaques malicieuses est ''quand un hôte devient suspect''.

Esparza [17] propose une solution qui se base sur la limitation du temps d'exécution de l'agent dans les hôtes. Il souligne son adéquation de l'intégrer à la solution des traces cryptographiques de Vigna. Dans notre approche nous qualifions un hôte d'un caractère suspect lorsqu'il manquera à l'obligation de délivrer son évidence de coopération et son information fournie est divergente dans le processus de vote. Alors, une négociation avec l'hôte s'impose afin de proclamer le caractère malicieux d'un hôte suspect. Par conséquent, le problème difficile de la détection des hôtes malicieux est remplacé par un simple processus de vérification de l'opportunité des explications de l'hôte. Nous avons l'ultime conviction que

plus de charges sur l'hôte réduira de manière significative la tentation d'altérer l'agent mobile.

Dans notre proposition, la collecte des évidences de coopération des hôtes est réalisée grâce à un protocole de non répudiation équitable. Par conséquent, les agents mobiles peuvent être distingués de manière asymétrique selon la confiance des hôtes visités. Les hôtes qui sont potentiellement d'un caractère suspect, sont ceux qui manqué à l'obligation de la délivrance des évidences de coopération pendant la collecte. Le propriétaire de l'agent maintiendra une classification des hôtes, utilisée ultérieurement lorsqu'il veut se plaindre. Les raisons principales du succès du protocole de la duplication et du vote étendu sont attribuées aux : (1) la disponibilité d'un même service offert par des hôtes indépendants affiliés à des organisations indépendantes. (2) du fait que les hôtes sont indépendants, les attaques collaboratives contre les agents mobiles sont écartées. (3) la pression exercée par la présence d'un réel potentiel de se plaindre dans le protocole proposé. Ainsi, nous pouvons affirmer qu'un résultat commun d'hôtes de confiance est fiable.

Le protocole proposé commence par collecter les évidences de coopération. L'absence d'une telle évidence d'un hôte est une condition nécessaire qui qualifie le caractère suspect. Contrairement, la présence de l'évidence de coopération d'un hôte est le symbole de confiance de l'hôte. Même si les agents dupliqués visiteront les hôtes selon la première planification, seuls les agents exécutés chez les hôtes de confiance participent dans le processus de vote. Les résultats divergents des autres agents sont simplement ignorés. Par contre, les résultats identiques au résultat du vote ne font qu'augmenter la crédibilité de ce résultat. Le protocole proposé est un protocole générique à plusieurs étapes. Ci-après est sa description:

1. Making a number of clones
2. Performing a fair non repudiation protocol without trusted third party
3. Classify hosts by cooperation evidence availability
4. Moving all clones
5. Each stage's vote is among information of clones visiting cooperative hosts
6. Classifying clones upon return, according to information handed out hosts ' class

7. Performing a vote among agents handing out information of cooperative hosts

8. Determining malicious hosts and trusted hosts with bizarre behavior

Algorithm 1: Proposed Protocol

La deuxième étape du protocole adopte un incontestable protocole de non répudiation équitable. En général, les protocoles de non répudiation supposent que les parties impliquées sont mutuellement suspectes. Plusieurs travaux traitent le problème du non répudiation. Les modèles de ISO/IEC13888 et de l'étude menée par [60] font intervenir une tierce partie de confiance dès lors qu'un problème arrive. Une deuxième catégorie de protocoles [61, 62] regroupe des protocoles dépendant du hardware, ou de la puissance de calcul des parties participantes. Nous avons choisi d'adapter le protocole proposé par [63], car il est générique et ne nécessite pas une tierce partie de confiance. Ainsi, le protocole générique de la collecte des évidences de coopération est présenté ci-après :

The agent owner determines the date D

1. Agent-Owner→Host-Target :Sign $_{\text{Agent-Owner}}$ (Move-Request, Agent-Owner,Host_Traget,D)
   The Host-Traget : Checks date
   i. Chooses n
   ii. Computes the singed f1,……,fn
2. Host-Targ →Agent-owner : Sign $_{\text{Host-Traget}}$ (fn(message), Host-Traget, Agent-Owner,D)
3. Agent-Owner→Sign $_{\text{agent-owner}}$(ack1)
4. .
5. .
6. .

2n. Host-Target→Agent-Owner :Sign $_{\text{Host-Traget}}$(f1(message).host-traget,agent-owner,D)

2n+1. Agent-Owner→Sign $_{\text{Agent-Owner}}$(ackn)

Algorithm 2: Evidences Collection Protocol

L'évidence de coopération est $Sign_{Host\text{-}Target} = \{Sign_i \mid i = 1,\ldots\ldots,n \text{ with } Sign_i = Sign_{Host\text{-}Target}(f_i(message, Host\text{-}Target, Agent\text{-}Owner, D)$. La probabilité pour que le propriétaire de l'agent ne transmette pas son acquittement et essaye de calculer précisément le message de la dernière étape est dépendante de la distribution géométrique utilisée lors du choix du nombre n. Dans ce cas, le hôte cible est supposé avoir transmis toute l'information nécessaire au calcul de l'évidence de coopération sans que le propriétaire de l'agent n'envoie son acquittement de numéro n. Cette dernière situation est très utile lorsque le ''principal'' Agent-Owner est une personne familière.

Dans la septième étape du protocole de la duplication et du vote étendu, le processus du vote est exécuté. Même si le vote est le mécanisme de validation de l'information retournée, il convient de dire que dans notre protocole, le vote édifiera les bases de la détection des hôtes malicieux. Les agents éligibles dans le vote sont ceux dont l'information proviendrait des hôtes de confiance. Le résultat partagé par une majorité d'hôtes de confiance est considéré comme fiable. L'information divergente au résultat du vote des autres hôtes est tout simplement ignorée. Cependant, à ce moment ces hôtes sont considérés des hôtes falsificateurs. Il est fort probable que ces hôtes sont malicieux. Ce comportement malveillant est vérifié avec la collaboration d'un arbitre, .i.e. un médiateur. Le prioritaire de l'agent alors contacte l'arbitre pour se plaindre. Le protocole générique de plainte est décrit ci-après :

/*Agen-owner enterning complaining state

i.e. some hosts are characterized by

A. absence of cooperation evidance

B. diverging with vote result */

1. 1. Agent-owner→adjudicator : $Sign_{agent\text{-}owner}$(move-time,move action,list suspicious result, list susupcious hosts,vote result,list of hosts vote)
2. Adjudicator→A vote host :$Sign_{adjudicator}$("give me :visiting agent identity & it's result but at move-time ")
3. Compose a check-list : (i.e., creteria model), list malicious hosts : = ø
4. For each suspicious Host :

   Adjudicator→Suspicious Host : $Sign_{adjudicator}$("explain your misbahavoir")

If(no replay or explanation not sound) then Add suspicious Host to list of maliciousHosts

5. For each malicious host :
    1. Send a warn , add warn to the log if none
    2. Advance malicious host log warn
    3. Threshold reached : revoke malicious host

Algorithm 3: Complaining Protocol

Dans la phase de plainte, le propriétaire de l'agent envoie à l'arbitre une preuve de base de la falsification. Cette preuve de base est un message signé, composé du temps de move, move-action, suspicious results, identities of potential malicious hosts, the majority Vote résult, and majority Vote trusted hosts list. Ensuite, l'arbitre construit la preuve de falsification correspondante, .i.e. Check-List ou Criteria Model. Pour accomplir cette tâche, l'arbitre examine avec précaution les conditions de l'exécution du protocole de non répudiation équitable. Il peut confirmer le déplacement des agents supposés falsifiés aux hôtes supposés malicieux. Dans ce cas, il interroge un hôte de confiance parmi les hôtes de confiance majoritaire dans le vote. L'assertion est d'autant vraie quand le hôte confirme l'arrivée d'un agent à l'instant du ''move''. Cela est convaincant car l'action de ''move'' est dotée de la propriété d'atomicité. Au même moment, l'arbitre est capable de valider le résultat du vote. Le procédé de validation collecte les résultats des agents migrateurs et proclame la validité du résultat en cas d'égalité. Ensuite, l'arbitre demande aux hôtes d'expliquer leurs comportements malveillants. Un hôte dont : (1) son évidence de coopération est absente. (2) son information diverge avec le résultat du vote, est déclaré malicieux. Plus tard, l'arbitre peut intervenir et avertir les hôtes malicieux car il a suffisamment d'information lui permettant de décider du ressort des explications fournies. L'explication est infondée, si elle n'arrive pas à justifier l'absence de l'évidence de coopération. Dérouter l'arbitre par une ancienne évidence de coopération est très tôt réfuté. Les évidences de coopération expirent à chaque fin d'une session de visite. Techniquement, une évidence de coopération est obsolète lorsque sa date est incompatible avec la date d'une évidence de coopération d'un hôte de confiance.

L'arbitre peut facilement être doté des compétences d'une autorité de sanction et révoquer les hôtes malicieux récidivistes. Contrôler l'historique des hôtes malicieux est extensible aux hôtes minoritaires dans le vote. Les hôtes minoritaires à répétition sont qualifiés d'un comportement bizarre, .i.e. étrange. Les identités de tels hôtes peuvent être disséminées sur le net.

5. Conclusion

Ce chapitre a défini les concepts nécessaires pour de modéliser les nouvelles versions des actions de clonage et de move. La spécificité de ce ''move'' est un peu particulière. Il déplace un ensemble d'agents à des emplacements différents en une seule action. La particularité de ces deux actions est qu'elles sont atomiques. Les deux actions sont définies comme une transformation de la plus faible pré-condition. A base de ces deux actions, le protocole de la duplication et de vote de Minsky est étendu afin d'incorporer un arbitre, .i.e. un médiateur. L'extension permet à l'arbitre de déterminer les hôtes malicieux et révoquer probablement ceux qui sont récidivistes. Au-delà, le protocole discerne les comportements bizarres des hôtes de confiance et il en capable de les disséminer.

Chapitre 4

Traces Cryptographiques & Limitation de Temps d'Exécution

1. Introduction

L'objectif des techniques de détection d'actes malicieux est d'identifier les modifications illégales du code, état, et le flux d'exécution d'un agent mobile. Tandis que, le code statique est facilement protégé par les signatures numériques, l'état et le flux d'exécution, par contre, nécessitent bien d'autres mécanismes. Plusieurs techniques de détection sont proposées dans la littérature, chacune se singularise à part de par ses mécanismes et principes. Parmi lesquelles on retrouve (1) la technique de ''state appraisal'' qui associe à un agent mobile une fonction d'appraisal. Lorsqu'un agent mobile arrive sur une nouvelle destination, la fonction d'appraisal est évaluée à base de l'état actuel de cet agent. En l'occurrence, la fonction d'appraisal vérifié, la validité de l'invariant de l'état. C'est ainsi que, les atteintes de l'état de l'agent sont détectées. (2) la technique basée sur les assertions protectives softwares offre une plus grande observabilité sous la forme d'images figées ''snapshots'' de l'agent. Ces dernières fournissent au propriétaire de l'agent (1) un moyen de déterminer le degré de confiance des résultats de l'agent (2) une information pour déboguer l'agent mobile (3) une habilité à vérifier les contraintes temps réel et (4) des moyens d'identification des systèmes déficients en ressources ''resource-deficient'', restrictifs en droits d'accès ''grant insufficient access rights'', ou malveillants ''tamper with agents''. L'approche proposée englobe (1) une méthodologie et des outils pour sélectionner et embarquer les assertions protectives dans le code de l'agent mobile (2) une manière automatique d'analyser l'information collectée. La technique des traces cryptographiques de Vigna, en est une autre méthode de détection des attaques. Dans cette technique, l'agent mobile crée des traces des instructions, modifiant son état, et qui dépendent d'entrée provenant des hôtes, et ce tout le long de son séjour dans le réseau. Chacun des hôtes garde et procède à la sauvegarde du hash des traces et des résultats chez l'hôte suivant. Enfin de parcours, si le propriétaire de l'agent mobile souhaiterait vérifier l'exécution de son agent mobile, il doit le ré exécuter à base des résultats des traces rapatriées. S'il arrive que la nouvelle exécution ne corresponde pas aux traces alors l'hôte en question est un tricheur. Les traces cryptographiques de Vigna détectent non seulement les attaques, mais elles démontrent le comportement malicieux de l'hôte. Enfin, la technique proposée par Esparza consiste à limiter le temps d'exécution de l'agent mobile dans les hôtes. Son idée traduit le fait que les hôtes malicieux nécessitent du temps pour analyser et modifier à leurs

avantages l'agent mobile. Le contrôle de temps d'exécution dans les hôtes permet de détecter les manipulations frauduleuses opérées sur l'agent durant son exécution. Il est vrai de constater, que les approches basées sur la détection des attaques et la démonstration du comportement malicieux des hôtes, prouvent la pertinence de disposer d'une autorité indépendante de sanction dans les systèmes à agents mobiles. Mais, conformément à notre vision du rôle de l'autorité indépendante de sanction, résumer ses compétences à une simple proclamation de verdicts a contribué considérablement à rendre les traces de Vigna ainsi que le protocole de limitation du temps d'exécution d'Esparza complexes, et sans impacts réels dans la pratique. Cependant, ces deux dernières techniques présentent quelques idées intéressantes et utiles à notre travail. Par conséquent, comme préalable à notre travail ce chapitre décrit en détails ces deux techniques en deux parties distinctes.

2. Traces Cryptographiques

Les traces cryptographiques de Vigna proposent un mécanisme de détection de n'importe quel type d'acte malveillant d'hôtes à l'encontre d'un agent mobile. Ces traces sont des ''logs'', i.e., des historiques d'opérations exécutées pendant le cycle de vie d'un agent. Ce mécanisme permet à un propriétaire après le retour de son agent d'un séjour dans le réseau, de vérifier si l'historique de l'exécution de l'agent est conforme à celle d'une exécution intègre.

2.1. Traces D'Exécution

Le mécanisme proposé suppose que les individus d'un système de cryptage ''principals'', particulièrement, les utilisateurs et les propriétaires des hôtes, disposent d'une clé publique et secrète utilisée dans le cryptage et dans les signatures numériques.

Un agent mobile est composé d'un segment de code p associé à un état d'exécution S^i, déterminé d'un point d'exécution spécifique i. L'état d'exécution est composé des données globales, la pile des appels, et le compteur du programme. Nous nous alignons sur l'hypothèse que le segment de code est un segment statique figé pendant tout le cycle de vie de l'agent. Nous signalons, que ce segment de code est composé d'instructions, qui sont blanches ou noires. Une instruction blanche est une instruction qui modifie l'état d'exécution de l'agent en ne faisant intervenir uniquement que les variables internes de l'agent. Une instruction noire par contre modifie l'état d'exécution de l'agent à base d'information reçu de l'environnement tel une opération de lecture.

Une trace T^p de l'exécution d'un programme p est composé d'une séquence de pairs (n,s), dont n représente un identificateur unique d'une instruction, et s une signature. Concernant une instruction noire, la signature contient les nouvelles valeurs correspondantes attribuées aux variables internes de l'agent. Par contre une signature d'une instruction blanche est toujours vide.

La suite de cette partie va préciser l'utilisation des traces d'exécution dans des situations de degré de complexité croissant, en commençant par l'évaluation à distance ''remote code execution'', en arrivant aux cas des agents mobiles. Signalons, que toutes les descriptions des protocoles suivants utilisent la notation suivante :

$X \xrightarrow{mi} Y : F_1, F_2, \ldots, F_n$ expliquant que le principal X envoi le message m_i au principal Y composé de champs $F_1, F_2, \ldots, F_n$.

2.1.1. Evaluation à Distance

L'évaluation à distance est un mécanisme qui permet à une application d'envoyer du code à un hôte distant pour l'exécuter. Plusieurs systèmes à agents mobiles tel Obliq et MO sont basés sur le mécanisme de l'évaluation à distance.

Un tel phénomène est décrit par l'expression :

$A \xrightarrow{m1} B : A_s(A, B, i_A, t_A, K_A(p), TTP)$.

Les deux premiers champs du message spécifient que le message destiné à B arrive de la source A. Quand B reçoit m_i, il utilise la clé publique de A pour vérifier la signature du message. Ainsi, B est assuré que l'origine du message est A et qu'il lui est destiné. Le troisième champ (i_A) est un identificateur unique dont le rôle est de protéger cette requête d'exécution des attaques de ''replay''. Le champ suivant (t_A) est une estampille inédite ''fresh''. Le champ qui vient juste après, représente le code à exécuter, crypté par la clé secrète de A. Enfin, le dernier champ est un identificateur de la tierce partie de confiance (i.e., TTP) responsable de régler les disputes d'actes d'un jeu inéquitable des ''principals''.

Le principal B est en mesure d'accepter ou de rejeter la requête de A suivant l'information du message. Dans les deux cas de figure, B répond par un message signé M dénotant le résultat de la décision. En général, le refus est motivé par un message d'erreurs. Par contre, l'acceptation est signalée par un accord définitif de B à exécuter p signifiant implicitement à A la nécessité d'envoyer sa clé publique K_A :

$B \xrightarrow{m2} A : B_s(B, A, i_A, H(m_1), M)$.

Lors de la réception du message, le principal A procède à sa validation. Entre autre, il s'assure que le message concerne la requête d'exécution identifié par i_A. Le protocole prend fin si M est un rejet. Autrement, le principal A envoie un message signé véhiculant la clé publique de A sous une forme crypté par la clé publique de B :

$A \xrightarrow{m3} B : A_s(A, B, i_A, B_P(K_A))$.

Lorsque B reçoit le message et après sa validation, il extrait la clé publique de A en utilisant sa clé secrète. Ensuite, le code p est décrypté en utilisant la clé publique de A pour enfin envoyer un message d'acquittement à A :

$B \xrightarrow{m4} A : B_s(B, A, i_A, H(m3))$.

Pendant l'exécution de p, le principal B produit la trace associée T^p_B. Comme convenu, la trace contient les identificateurs des instructions exécutées ainsi que les signatures des instructions noires.

Au moment de la halte de l'exécution de p, le principal B envoie un message signé à A composé de l'état final d'exécution S_B cryptée par la clé aléatoire K_B et le checksum de la trace produite pendant l'exécution et enfin une estampille t_B :

$B \xrightarrow{m5} A : B_s(B, A, i_A, K_B(S_B), H(T^p_B), t_B)$.

Lorsque le principal A reçoit le message, il répond par un message d'acquittement, lequel est implicitement simulé à une demande de la clé de l'extraction des résultats de l'exécution :

$A \xrightarrow{m6} B : A_s(A, B, i_A, H(m_5))$.

Ce message assure le principal B de l'accord du principal A à payer les services utilisés par le code mobile au cas où l'exécution est correcte. La réponse du principal B comporte la clé de décryptage des résultats, cryptée par la clé publique de A :

$B \xrightarrow{m7} A : B_s(B, A, i_A, A_p(K_B))$.

Après avoir décrypté les résultats de l'exécution, si pour une raison ou une autre le principal A suspect le principal B d'être malveillant, il peut demander à B de produire la trace. Du moment que le message m_5 existe, le principal B est dans l'incapacité de refuser une telle demande. Après la réception de la trace complète de B, le principal A vérifie sa

concordance à celle qui se trouve dans le message m_5. Enfin, le principal A procède à la validation de l'exécution de *p* en accord avec la trace T^p_B. Cela se traduit par l'exécution étape par étape, et à chaque étape l'identificateur de l'instruction courante est comparé avec celui du même niveau contenu dans la trace. A la rencontre d'une instruction noire, la valeur de l'entrée est extraite de la signature correspondante. Plusieurs résultats intéressants déterminant des comportements non sains des deux partenaires du processus de validation sont envisageables. L'idée de base est de rechercher les contradictions éventuelles entre les traces et l'exécution simulée. Le tableau suivant récapitule les cas de figures qui sont décelables :

B tricheur	A tricheur
1. Trace instructions blanches ≠ Trace flux exécution simulée	1. refus de payer les charges
2. Disparité état final S_B et état final de simulation	
3. Pas de disparité et charges non conforme	

Table 3 : Cases of Malicious Acts of both Principals

Selon le cas, le principal concerné est en mesure de fournir une preuve irréfutable à la tierce partie de confiance de l'acte malveillant de son partenaire. Cela suppose que les deux principales participantes sont de bonne volonté (playing fair) .i.e. exécutant comme prévu le protocole. Dans le cas contraire, la tierce partie de confiance peut être impliquée afin d'obliger (coercition) les deux partenaires à jouer de manière équitable (fair play).

2.1.2. Agent Mobile

Les traces d'exécution d'un agent mobile sont plus complexes à établir. Il y a lieu d'en disposer d'informations supplémentaires permettant d'identifier clairement toute exécution anormale de l'agent sur n'importe quel hôte que ce soit. Isoler l'exécution anormale dans un parcours composé de plusieurs hôtes requiert une capacité de validation pré et post mortem de la trace d'exécution aux frontières d'une migration vers un prochain hôte de l'itinéraire. C'est pour cette raison, que chaque hôte garde non seulement une copie de sa propre trace d'exécution mais sauvegarde aussi chez son prochain hôte, le hash de sa trace et de l'état final d'exécution correspondant de l'agent. De plus, une information statique sur l'agent appelée ''agent token'', présente tout le long de son parcours atteste de l'intégrité de sa partie statique

du code et de l'identité de son site d'origine. Ce mécanisme augmenté comme dans le cas des traces d'exécution au niveau d'une évaluation à distance détectent toute tricherie d'un hôte dont la simulation de l'exécution fait ressortir une contradiction avec sa trace lors d'une vérification finale éventuelle. Bien sûr cette vérification finale ne peut avoir lieu qu'après le retour de l'agent mobile à son site d'origine.

Ici, on considère un scenario où l'agent mobile démarre d'un site d'origine et puis se déplace d'un site à un autre avec le but d'accomplir une tâche particulière. Admettons que, l'agent démarre du site d'origine A, et à un moment de son exécution, il demande de migrer vers le site B. Par conséquent, son code p et son état S_A sont transférés au site B. Le site A envoie au site B le message signé suivant :

$$A : \ m_1 \rightarrow \ B : A_s(A, B, K_A(p, S_A), A_s(A, i_A, t_A, H(p), TTP)).$$

Les deux premier champs assument que le message est envoyé de A à B. Alors que, le champ suivant contient le code (p) et son état initial (S_A) chiffrés à l'aide de la clé aléatoire choisit par A. Le quatrième champ est ce qui est appelé ''agent token'', une information statique de l'agent qui sera utiliser dans les sites visités prochainement. Il contient l'identité de A, son identificateur i_A, une estampille t_A de son transfert vers B, la valeur du hash de son code, et enfin l'identité de la tierce partie de confiance. Le jeton ''token'' est signé par A.

Quand le message m_1 arrive chez B, ce dernier utilise la clé publique de A et procède à la vérification du message et du ''agent token''. B est capable de refuser ou d'accepter d'exécuter l'agent selon l'information du message comme dans le cas de l'évaluation à distance. Indépendamment du cas, B envoi à A le message signé suivant :

$$B : \ m_2 \ \rightarrow A : B_s(B, A, i_A, H(m_1), M).$$

Le principal A débute par valider le message de B et examine M. Le protocole est terminé dans le cas où M est un refus. Le cas contraire M est la confirmation que B exécutera l'agent et il a besoin de la clé K_A. A envoi à B le message suivant :

$$A : m_3 \rightarrow \ B : A_s(A, B, i_A, B_p(K_A)).$$

Le principal B procède à la vérification de la validité du message, à l'extraction de la clé de A en utilisant sa clé secrète, et déchiffre le code et l'état de l'agent. Ensuite, il envoie un acquittement :

$B : m_4 \rightarrow A : B_s(B, A, i_A, H(m_3))$,

et lance l'exécution de l'agent. Lorsque l'agent demande son transfert vers un autre site C, le site B arrête l'exécution de l'agent et envoie le site C deux messages consécutifs signés :

$B : m_5 \rightarrow C : B_s(B, C, agent_A, H(T^p_B), H(S_B), t_B)$,

$B : m'_5 \rightarrow C : B_s(K_B(p, S_B), H(m_5))$.

Le premier message contient les noms de la source et de la destination, le token de l'agent, la valeur hash de la trace correspondante à l'exécution de l'agent chez le site B, la valeur hash de l'état actuel S_B, et une estampille t_B. Le second message contient le code et l'état actuel de l'agent, chiffrés par la clé aléatoire K_B, et un hash du message précédent. Le site C vérifie le site de départ de l'agent et la validité du message m_5 et répond à m_5, m'_5 par le message suivant :

$C : m_6 \rightarrow B : C_s(C, B, i_A, H(m_5, m'_5), M)$.

Selon que M est un refus ou une acceptation, le site B relance l'exécution de l'agent avec un message d'erreurs ou envoie sa clé de déchiffrage à C sous la forme du message suivant :

$B : m_7 \rightarrow C : B_s(B, C, i_A, C_p(K_B))$.

Le site calcul H(*P*) et puis sa valeur à celle qui se trouve dans le token de l'agent. Ensuite, il compare le hash de S_B et à la valeur correspondante du message m_5. Dans le cas d'un transfert intègre, le site C envoie un acquittement et termine le transfert :

$C : m_8 \rightarrow B : C_s(C, B, i_A, H(m_7))$.

Le même procédé est répété dans les sauts suivants jusqu'au où l'agent termine son itinéraire. Alors, le dernier site contact le site de départ afin de lui délivrer l'état final de l'agent.

Le site d'origine et après avoir décrypté les résultats de l'exécution, si pour une raison ou une autre, il suspect des sites d'êtres malveillants, il peut commencer à demander à B de produire la trace. Du moment que le message m_2 existe, le principal B est dans l'incapacité de refuser une telle demande. Après la réception de la trace complète de B, le principal d'origine lance la simulation à base de la trace reçue. Lorsque la simulation atteint l'instruction du transfert au site C, l'état simulée de l'agent est noté S'_B. Le site A demande au site C sa trace

et une copie du message m_5. L'existence du message m_7 le contraint de faire de la sorte. Après extraction, $H(S'_B)$ est comparé à $H(S_B)$ du message m_5 ainsi que l'intégrité de la trace fournit par le site B. Plusieurs résultats intéressants déterminant des comportements non sains de deux partenaires du processus de validation sont envisageables. L'idée de base est de rechercher les contradictions éventuelles entre les traces et l'exécution simulée. Le processus est répété pour chaque site de l'itinéraire et les conclusions singulières de chaque site sont semblables à ceux de l'évaluation à distance.

3. Limitation du Temps d'Exécution

Esparza introduit l'idée de limiter le temps d'exécution des agents mobiles chez les hôtes. Les hôtes malicieux ont nécessairement besoin d'un temps pour analyser et modifier un agent afin de profiter d'un gain possible. Contrôler le temps d'exécution dans les hôtes permet de détecter les attaques de manipulation des hôtes malicieux lors de l'exécution de l'agent. Dans la technique proposée par Esparza, chaque hôte garde une trace du temps d'arrivée et de fin d'exécution d'un agent calculés par rapport à un temps de référence général. Ces deux temps ainsi que les résultats sont envoyés à l'hôte d'origine dans l'objectif de vérifier les incohérences des temps d'exécution et de transmission. Dans le cas de détection d'hôtes suspects, leurs résultats sont alors écartés. Concrètement, le procédé est proposé sous la forme d'un protocole de plusieurs phases.

3.1 Protocole

Le protocole suppose que tous les messages échangés sont signés à la source. L'objectif est d'éviter les attaques de répudiation. Le protocole est divisé en trois différentes parties : la partie configuration, la partie messagerie de l'agent, et la partie de vérification.

3.1.1 Configuration

Dans cette partie de configuration du protocole, l'hôte d'origine fournit aux hôtes de l'itinéraire de l'agent mobile les paramètres dont ils ont besoin dans les parties suivantes du protocole. Les étapes de cette partie du protocole sont expliquées ci-après :

1. L'hôte d'origine prend le rôle d'un client dans ce protocole. Avant d'envoyer l'agent, cet hôte doit nécessairement initialiser le système et accomplir les tâches suivantes :

a) Générer un référentiel général de temps noté T_0. Les temps d'arrivée et fin d'exécution de tous les hôtes sont calculés à base de T_0.

b) Générer aussi une clé de session K_s, utilisé pour crypter l'agent. On peut utiliser un algorithme Symétrique de cryptage tel DES.

c) S'accorder un temps de tolérance en concordance avec le niveau de sécurité du système. Correspond à une visibilité d'un retard imprévisible.

d) Amorcer le référentiel général de temps T_0. L'unité de mesure de temps doit être assez fine pour garantir un calcul correct du temps d'exécution et de transmission.

e) Envoyer un message séparé contenant K_s et T_0 à chaque hôte. Le message est crypté par la clé publique de chaque hôte afin d'établir un canal sécurisé de communication.

2. Prenons le rôle d'un server, chaque hôte d'exécution accomplit les tâches suivantes :

a) Reçevoir le message de l'hôte d'origine et amorce immédiatement le compteur de référence de temps. Ce temps est noté T_i où i reflète la position dans l'itinéraire.

b) Envoyer un message d'acquittement à l'hôte d'origine lui signifiant son accord d'exécuter l'agent car il a suffisamment de ressources. dans le cas contraire il observe le silence.

3. L'hôte d'origine accomplit les tâches suivantes :

a) Rassembler les acquittements de hôtes, à défaut d'un ou plusieurs acquittements, une reconfiguration de l'itinéraire est effectuée en faisant intervenir que les hôtes avec acquittements. Les hôtes sans acquittements sont considérés non disponibles. La reconfiguration de l'itinéraire commence par l'envoi au reste des hôtes une nouvelle clé de session. Ce processus ne s'arrête que lorsque tous les acquittements de l'itinéraire sont rassemblés.

b) Activer un évaluateur du temps de transmission (TTE) pour calculer le délai de transmission entre chaque hôte de l'itinéraire et l'hôte d'origine. Il inclut aussi les effets du délai de propagation. Le délai de transmission est noté par $T_{delay\_o\_i}$ de l'hôte de la position i de l'itinéraire. Sa valeur est utilisée pour estimer le décalage entre le temps de référence général T_0 et le temps de référence de chaque hôte T_i. La relation suivante est attestée :

$$T_i \equiv T_0 + T_{delay\_o\_i}$$

c) Activer un estimateur du temps d'exécution (ETE). L'objectif est de calculer le temps nécessaire à l'exécution complète de l'agent par chaque hôte. Ce temps estimé à la position i est noté $T_{exec\text{-}i}$.

d) Estimer le temps de cryptage des résultats et puis envoyer l'agent. Ce temps est une partie intégrante de $T_{exec\text{-}i}$ et il est noté $T_{send\text{-}i}$.

Le schéma suivant résume l'essentiel du travail accompli dans la phase de configuration.

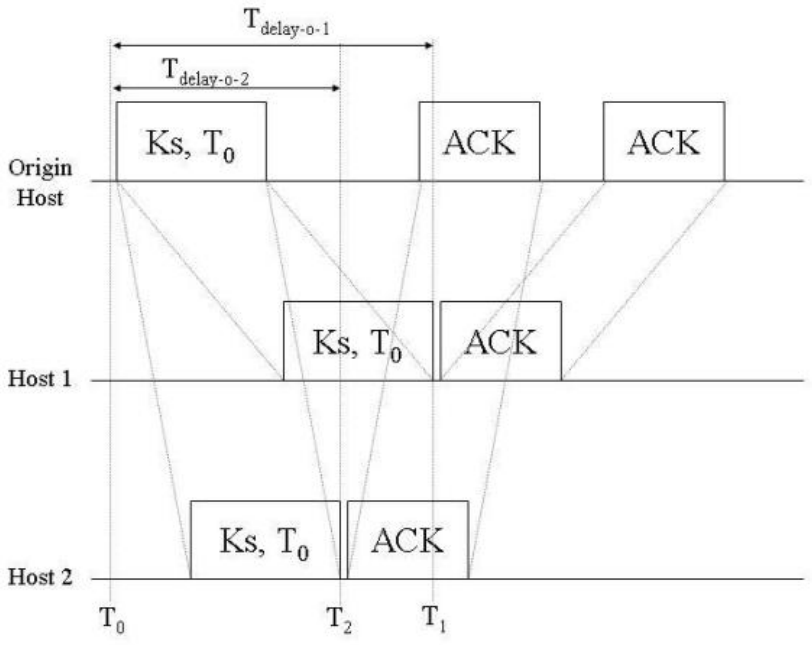

Figure 5 : Configuration part

3.1.2 Phase d'envoi de l'agent

Juste après la phase de configuration, l'agent mobile commence sa journée. Les traces du temps d'arrivée et du temps de finalisation sont sauvegardées avec les résultats de l'exécution à chaque saut de l'itinéraire. Les étapes du protocole de cette phase sont expliquées ci-après :

4. L'hôte d'origine accompli les tâches suivantes :
    a) La TTE calcule le délai de transmission entre deux hôtes consécutifs. Concernant le hôte de la position i et le hôte de la position i+1, ce délai est noté $T_{trans\text{-}i\text{-}i+1}$
    b) L'agent est crypté avec la clé de session
    c) Enfin, l'agent est envoyé vers le premier hôte de l'itinéraire. Le temps de départ de l'agent noté $T_{final\text{-}o}$ utile est sauvegardé.
5. Le premier hôte de l'itinéraire recevra l'agent et accompli les tâches suivantes :
    a) Sauvegarde le temps d'arrivée relativement à T_1 noté $T_{arriv\text{-}1}$
    b) Décrypter le code en utilisant K_s
    c) Exécuter le code
    d) Sauvegarder le temps de finalisation relativement à T_1 noté $T_{final\text{-}1}$
    e) Crypter à l'aide de la clé publique de l'hôte d'origine les résultats et les deux temps précédents.
    f) Envoyer l'agent à l'hôte suivant de l'itinéraire. Ces deux dernières tâches sont nécessaires pour un envoi sécurisé. Il faut juste rappeler que le temps estimé de cette opération est calculé par ETE et il noté $T_{send\text{-}i}$
6. Chaque hôte de l'itinéraire essai de faire les mêmes tâches décrites au paragraphe 5. Finalement, le dernier hôte envoie tous les résultats à l'hôte d'origine.

7. L'hôte d'origine :

a) Reçoit l'agent et sauvegarde son temps d'arrivée $T_{arriv\text{-}o}$.

b) Décrypte les résultats ainsi que les temps d'arrivée et de finalisation de chaque hôte.

Le schéma suivant résume l'essentiel du travail accompli dans la phase de l'envoi de l'agent.

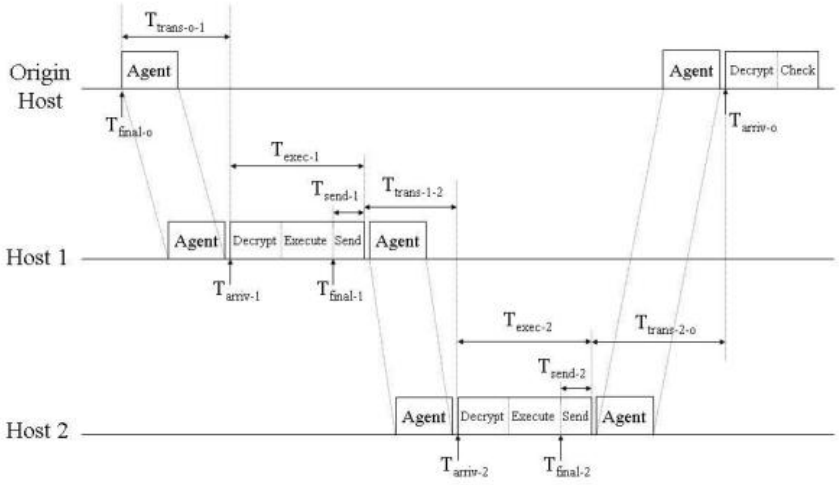

Figure 6 : Agent sending phase

3.1.3 Partie Vérification

L'hôte d'origine vérifie l'intégrité des temps d'exécution des hôtes de l'itinéraire en trois étapes :

- Un hôte est suspecté d'avoir entrepris des actes malicieux si son temps réel d'exécution est supérieur au temps estimé d'exécution par seuil qui dépasse celui de la tolérance. Les hôtes honnêtes doivent satisfaire l'expression suivante :

  $(T_{final\text{-}i} - T_{arriv\text{-}i} + T_{send\text{-}i}) <= T_{exec\text{-}i} + \text{Tolérence}$

  Le système est d'autant mieux sécurisé lorsque la valeur de la tolérance est plus petite.

- Le temps de départ d'un hôte ne peut être plus grand que le temps d'arrivée de l'hôte suivant. Les hôtes honnêtes doivent satisfaire l'expression suivante :

  $T_{arriv\text{-}i+1} \mid regardsT_{i+1} \geq T_{final\text{-}i} \mid regardsT_i + T_{send\text{-}i}$

  Les temps de référence des hôtes étant différents, il est nécessaire de réajuster les entités à comparer pour qu'elles deviennent relatives à T_o. Cela est réalisé en soustrayant le temps de transmission de l'hôte d'origine. Ainsi, l'expression est réécrite de la façon suivante :

  $$T_{arriv\text{-}i+1} - T_{delay\text{-}o\text{-}i+1} \geq T_{final\text{-}i} - T_{delay\text{-}o\text{-}i} + T_{send\text{-}i}$$

  Quand l'expression est non satisfaite, les deux hôtes sont suspects. La discrimination entre les deux hôtes dans ce cas est impossible.

- La différence entre les deux termes doit être similaire au temps de transmission. Les hôtes honnêtes doivent satisfaire l'expression suivante :

  $|[(T_{arriv\text{-}i+1} - T_{delay\text{-}o\text{-}i+1}) - (T_{final\text{-}I} - T_{delay\text{-}o\text{-}i} + T_{send\text{-}i})] - T_{trans\text{-}i\text{-}i+1} \mid <=$ tolerance.

  Quand l'expression est non satisfaite, les deux hôtes sont suspects. La discrimination entre les deux hôtes dans ce cas est impossible.

Lorsque l'hôte d'origine constate des incohérences dans les temps vérifiés, il peut décider d'ignorer les résultats des hôtes suspects ou lancer une nouvelle exécution de l'agent en supprimant tous les hôtes suspects de l'itinéraire.

Selon Esparza le protocole décrit ci-dessus au-delà de sa détection d'attaques perpétrées par un seul hôte, il est capable aussi de détecter quelques attaques de collusion.

3.2 Attaques

Les attaques tentent de satisfaire de manière illégale les vérifications du temps dans le protocole en falsifiant les temps d'arrivée et de finalisation. On admet que les temps estimés

par ETE et TTE sont très proches de la réalité afin d'entraver l'analyse et la modification de l'agent.

3.2.1 Attaques d'un seul hôte

Le protocole détecte toute attaque originaire d'un seul hôte. Ces attaques sont listées ci-après :

- L'hôte malicieux ne ment pas à propos de son temps d'exécution, pendant lequel l'hôte a analysé et a modifié le code ou l'état de l'agent. Cette attaque est facilement détectée car la première vérification du temps attestera que le temps d'exécution est plus grand que le temps d'exécution estimé par l'ETE.

- L'hôte malicieux ment à propos de ces temps. Son objectif est de blâmer à un autre hôte. Dans ce cas deux possibilités sont de mise :

  -L'hôte malicieux ment à propos de temps d'arrivée. Alors, il sauvegarde une valeur plus du temps d'arrivée. au cas où le temps d'arrivée est augmenté par un nombre qui dépasse la tolérance, la troisième vérification du temps ne sera pas satisfaite et par conséquent l'attaque est détectée.

  -L'hôte malicieux ment à propos du temps de finalisation. Il sauvegarde une valeur inférieure au temps de finalisation. En cas où le temps de finalisation est réduit d'un nombre plus grand que la tolérance, la troisième vérification du temps ne sera pas satisfaite et conséquence l'attaque est détectée.

3.2.2 Attaques de collusion

Sous une certaine condition, le protocole est capable de détecter quelques attaques de collusion**,** parmi lesquelles on retrouve, la transmission du code original, la transmission du code modifié. Dans ce cas, l'exécution de l'agent réel est substituée par celle d'une copie déjà analysée et modifiée. Il semblerait que l'analyse et la modification de l'agent doit se faire par un hôte en dehors des limites calculées par le protocole i.e., en un temps mort. C'est pour cette raison que l'attaque de la transmission du code modifié est vraisemblablement improbable à moins que l'hôte ne soit éligible à commettre un suicide. En tout cas, la condition de détection de telles attaques coïncide avec le fait que le temps de l'exécution de l'agent substitut est plus grand que le temps estimé de l'exécution de la copie d'origine.

4. Conclusion

Les deux techniques présentées dans ce chapitre sont des mécanismes de détection d'actes malicieux des hôtes. Les traces cryptographiques de Vigna sont robustes. Elles permettent la détection de manipulation frauduleuse du code, de l'état, et du flux d'exécution d'un agent mobile. Quant à la technique de limitation du temps d'exécution d'Espaza, elle permet de détecter non seulement les comportements malicieux d'hôtes individuels, mais aussi quelques attaques de collusion. Plusieurs désavantages de ces deux techniques sont rapportés dans la littérature. En résumé, elles sont répertoriées dans la catégorie des techniques non réalistes en vue de la complexité de leur mise en pratique. Néanmoins, la notion des instructions noires des traces cryptographiques ainsi que le fait de noter que ''Les hôtes malicieux ont nécessairement besoin d'un temps pour analyser et modifier un agent et ce afin de profiter d'un gain possible'' sont deux idées émanant de ces deux techniques et qui sont des idées très intéressantes. Ces dernières sont exploitées d'une autre manière plus originale afin d'asseoir le développement d'une nouvelle approche de détection des hôtes malicieux. La nouvelle approche a le privilège d'être robuste et facile à implémenter i.e., d'un faible coût. Le chapitre suivant va traiter en détail les fondements et les protocoles qui en découlent de cette nouvelle approche.

Chapitre 5

Nouveau Protocole de Détection des Hôtes Malicieux

1. Introduction

Les techniques dont l'aptitude est de rendre les manipulations frauduleuses des agents impossible ou plus difficiles tels les techniques ‘’Time Limited Black Box’’, cryptographie mobiles sont moins viables que celles dont l'effort cherche à déterminer l'occurrence des actes malicieux des hôtes [11]. Nous adhérons à la thèse que la survivance du propriétaire de l'agent est fortement compromise lorsque l'information retournée est incorrecte où elle arrive en retard. Ceci est vrai car si le propriétaire de l'agent est induit en erreur de manière délibéré de par les informations de son agent, le propriétaire est certainement sous un assaut d'information. Lorsque cet assaut est contrarié, la survivance du propriétaire de l'agent est bien évidemment augmentée [11].

La nouvelle technique proposée s'inscrit dans le cadre général des mécanismes de détection d'actes malicieux des hôtes. Elle est robuste et simple à implémenter. Elle est capable de détecter les attaques et de démontrer le comportement malicieux des hôtes. Quelques principes de base sont à l'origine des politiques et mécanismes développés dans cette technique : (1) les décisions critiques d'un agent mobile sont prises au niveau d'hôtes de confiance (2) le flux d'exécution de l'agent est typique, i.e. standard dans les différents hôtes. L'infrastructure sur laquelle la technique proposée doit être déployée est l'une des plus traditionnelles dans le contexte des agents mobiles. En réalité, la technique est un processus incrémental subdivisé en deux grandes phases. Dans sa première phase, le processus s'efforce de déterminer le caractère suspect d'un hôte. La modélisation de ce caractère répond au souci de surpasser le temps d'exécution du modèle de référence d'exécution. Naturellement, le modèle de référence d'exécution n'est que le temps d'exécution de l'agent chez le premier hôte de l'itinéraire. Les temps des exécutions sont calculés tout en ignorant le temps des opérations des entrées sorties qualifiées par Vigna d'instructions noires. Dans sa deuxième phase, la technique est capable d'infirmer ou de confirmer et de manière progressive sans aucune ambiguïté le comportement malicieux de l'hôte en discriminant la vérité. Ce mécanisme exploite le principe lié aux ressources qui sont ‘’scarces’’ tel les cycles du processeur et la mémoire. Ce principe dû à Barak [65] atteste que ‘’ *n'importe quel serveur dont la charge de travail est proportionnelle à la taille du système, est destiné à être englouti, une fois cette taille dépasse une certaine limite’’*. Par conséquent, le caractère stable de la

vérité est utilisé afin de distinguer entre hôte malveillant et honnête. La discrimination de la vérité chez un hôte malicieux devrait utiliser une approche qui requiert une utilisation massive de l'espace de stockage et du temps du processeur. L'objectif est de surcharger l'hôte, le conduisant rapidement au stade d'un état où l'hôte est englouti. Nous estimons ainsi que l'hôte malicieux, et avant d'arriver dans l'état englouti se contredit à cause d'une carence dans les ressources ''scarces''.

Ce chapitre est organisé en six sections. Hormis la section introduction, dans la deuxième section, quelques travaux connexes sont décrits afin de montrer l'intérêt de notre approche. La troisième section explique la contribution majeure de cette approche. La section quatre développe dans un premier stade, un protocole de détection du caractère faible de suspicion des hôtes, ensuite elle explique les principes de la médiation, définit la proposition du protocole, et enfin propose des solutions aux questions pratiques soulevées. Afin de consolider le protocole proposé, la dernière section est consacrée complètement à la validation expérimentale de l'efficacité de la partie dévouée à la détection du caractère faible de suspicion des hôtes en utilisant une approche d'analyse statistique. Nous terminons enfin par une conclusion.

2. Les Travaux Connexes

Dans sa partie de détection du caractère suspect des hôtes, la technique proposée partage les idées clés des schémas présentés dans Vigna et Esparza [14, 17]. Cependant, elle procède différemment.

Vigna a privilégié le chemin de construction d'état final et initial figés à chaque saut de l'agent mobile. L'idée est concrétisée via une redondance des traces intermédiaires hachées et le maintien des règles du ''fair play''. L'information principale des traces intermédiaires est la valeur des instructions noires. Par conséquent, la vérification d'actes malveillants se fait à travers la simulation de l'exécution de l'agent en utilisant maintenant les traces figées. Toute disparité entre l'exécution d'un hôte et son exécution simulée confirme le caractère malicieux de l'hôte (i.e., hôte tricheur). Le coût de l'efficacité du protocole de Vigna est une demande excessive en mémoire, des phases complexes pour assurer la non répudiation (i.e., fair play) et une réexécution simulée de l'agent.

Esparza a pris une autre direction. Son idée clé est de limiter le temps d'exécution de l'agent mobile dans chaque hôte. Au départ, le protocole définit un référentiel de temps

général. Du moment que, la synchronisation des horloges des hôtes est impossible, un évaluateur du temps de transmission est activé afin d'estimer le délai de transmission entre hôtes consécutifs dans l'itinéraire. De même, un estimateur du temps d'exécution est nécessaire pour calculer le temps d'exécution dans chaque hôte. Selon ce protocole, les temps réels d'exécution et de transmission d'un hôte honnête sont confinés respectivement dans les limites :

1. du temps d'exécution estimé,
2. de la différence entre le temps d'arrivée chez le prochain hôte et le temps de finalisation de l'actuel l'hôte.

Evidemment, le temps absolu de finalisation d'un hôte est toujours inférieur au temps absolu d'arrivée chez son successeur. Le point crucial du protocole se situe dans ses composants d'estimation des temps. Quand l'estimation est statique, les valeurs estimées ne sont d'autant fiables que lorsque elles sont extraites de par une pré simulation massive. Une estimation réelle sophistiquée du temps engendre de ''l'overhead'' et une connexion permanente avec l'hôte d'origine, et ainsi, la propriété ''off-line'' est non respectée.

Nous nous intéressons aussi dans notre approche aux instructions noires et au temps d'exécution dans chaque hôte. Cependant, contrairement aux travaux de Vigna et Esparza :

1. Nous considérons que les instructions noires sont la source des irrégularités dans le temps d'exécution. Par conséquent, le temps d'exécution est arrangé en ignorant le temps consacré à ces instructions. Ceci permettra de rendre la puissance de calcul d'un hôte proportionnel à sa charge.
2. Nous écartons le besoin de recours à un référentiel général de temps et des composants de l'estimation. Nous considérons le premier hôte de l'itinéraire comme le modèle de référence de l'exécution. En réalité, même quand le premier hôte est un tricheur, notre protocole est capable de le détecter. Notre protocole n'a besoin que d'un temps signé et de la charge du travail de chaque hôte à l'arrivée et au départ de l'agent et aux frontières des instructions noires. Nous constatons qu'il n'y a nul besoin d'une connexion permanente avec le

propriétaire de l'agent pendant tout le long de son itinéraire. L'avantage est la préservation de la propriété ''off-line''.

3. Contributions Majeures

La stratégie proposée appartient évidement à la catégorie des techniques de détection des actes malveillants des hôtes. A notre connaissance, il n'y a aucune solution proposée dans la littérature qui a pris cette direction de recherche et d'exploration du problème des hôtes malicieux. A côté du mécanisme d'établissement de la vérité, notre solution n'entrave pas la flexibilité et la puissance du paradigme des agents mobiles. L'agent mobile n'a nul besoin d'inter agir avec sa base pendant tout son séjour. Aussi, ni des traces des états intermédiaires d'exécution ni d'autres difficultés de développement des agents mobiles ne sont requises. Dans notre solution, nous étendons le besoin de fournir à la tierce partie de confiance une preuve concrète d'actes malicieux. Les compétences de la tierce partie de confiance sont revues à la hausse. Par conséquent, un protocole d'un coût faible est dérivé. Son rôle est de détecter le caractère suspect des hôtes, basé sur les irrégularités observées au niveau des temps d'exécution. Ces temps sont réajustés par le processus de médiation de la tierce partie de confiance, afin de refléter le facteur de la puissance et la charge référentielles de calcul des hôtes, ce qui donnera une meilleure précision. Le rôle de la tierce partie de confiance dans l'analyse est plus élaboré. Quand une plus grande confidence est gagnée dans la revendication, l'analyse se termine par l'activation du processus de l'établissement de la vérité. Dans le cas contraire, quand les revendications ne sont que des simples allégations, le processus s'arrête. Dans le processus de l'établissement de la vérité, une tentative d'inonder le hôte suspect par des interrogations sous la forme de questions est adoptée. Cependant, le processus de l'interrogation ne doit pas être aveugle, i.e. inonder les deux parties de l'interrogation. La discrimination est possible si l'inondation est une tâche pivotée par l'information pertinente de l'hôte suspect. Par conséquent, la tierce partie de confiance peut poursuivre normalement son travail tant qu'elle dispose de moyens pour gérer les questions/réponses et basculer vers le calcul à base de moyennes dans le cas contraire.

4. Nouvelle Stratégie de Traitement des Hôtes Malicieux

Nous proposons une nouvelle stratégie de traitement des hôtes malicieux. La stratégie définie ainsi a deux phases. Pendant la première phase, nous allons développer un protocole qui préserve la propriété ''off-line'' de détection du faible caractère de suspicion d'un hôte. Le protocole rend la puissance de calcul d'un hôte, proportionnelle à sa charge. Cet objectif

est atteint en ignorant le temps consacré à l'acquisition des données externes, i.e. les résultats des instructions noires. Dans son étape finale, le protocole recherche les incompatibilités possibles parmi les fragments de temps calculés dans chaque hôte relativement à ceux du premier hôte de l'itinéraire. Ensuite, ces incompatibilités sont envoyées à la tierce partie de confiance pour une évaluation en profondeur. Quand une meilleure confiance est gagnée dans la revendication, l'activité de l'établissement de la vérité prend la main et termine le travail. Son objectif est de pousser l'hôte suspect à la limite de ses pleines capacités. Nous espérons créer, une fois la frontière d'une gestion pratique des ressources ''scarces'' de l'hôte suspect dépassée, une situation où cet hôte est complètement en état de désarroi. L'état de désarroi de l'hôte, illustré par une incompatibilité parmi les réponses de son ''log'' est considéré la preuve de confirmation d'actes malicieux. La figure suivante schématise l'exécution d'un agent mobile chez un hôte, élément principal de compréhension du fonctionnement de cette stratégie.

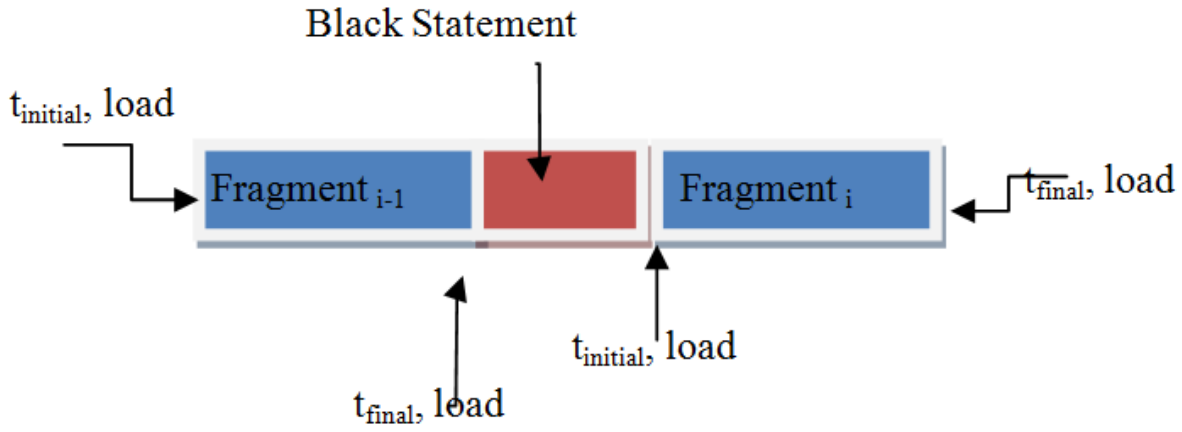

Figure 7: Agent's Code Fragmentation

4.1 Protocole de Détection du Caractère Suspect

Notre protocole ne nécessite qu'un simple modèle du paradigme des agents mobiles. Ainsi, la technologie sous-jacente de l'implémentation du modèle de l'agent mobile fournit des mécanismes cryptographiques, garantissant la confidentialité et l'intégrité de l'agent uniquement pendant son transit. Il faut rajouter à cela la communication à la tierce partie de confiance de par chaque hôte de sa puissance de calcul référentielle et la charge correspondante du travail. La communication doit avoir lieu avant l'entrée en service de l'hôte en question.

Nous considérons un scénario où l'agent mobile chargé d'une tâche particulière est migrant, .i.e. ''roving'', d'un hôte à un autre à partir de sa base. Eventuellement, l'agent mobile termine sa journée et les résultats sont délivrés au propriétaire de l'agent à la base. Juste après, le propriétaire de l'agent au niveau de la base commence la vérification des actes suspects et transmis et dans le cas positif ces derniers à des investigations détaillées chez le service de médiation, .i.e. la tierce partie de confiance. Nous expliquons les étapes du protocole dans chaque phase toute en décrivant les tâches accomplies au niveau de l'hôte de base ou de chaque hôte d'exécution impliqué. Les étapes du protocole sont expliquées ci-après :

L'hôte d'origine, i.e. la base procède au cryptage de l'agent mobile avec la clé publique du premier hôte de l'itinéraire et le véhicule à destination.

1. A la réception de l'agent mobile, l'hôte procède à son décryptage et sauvegarde dans son état le temps d'arrivée et la charge avant de lancer son exécution. Lorsque l'exécution atteint la frontière d'une instruction noire, l'hôte et de nouveau sauvegarde le temps courant et la charge du travail au niveau de l'état de l'agent. La même tâche est effectuée à la fin d'une instruction noire, et aussi avant qu'un nouveau déplacement de l'agent mobile à un prochain hôte ne soit initier. A la rencontre d'une instruction de saut, l'hôte actuel procède au cryptage de l'agent mobile avec la clé publique de l'hôte prochain et l'envoie à destination En fin de mission l'agent mobile retourne chez le hôte d'origine.

2. L'hôte d'origine procède au décryptage de l'agent mobile et lance la procédure de vérification des actes suspects. Comme nous l'avons déjà expliqué, le premier hôte est considéré comme le modèle de référence de temps. Par

conséquent, les fragments de temps des hôtes suivants sont réajustés afin de refléter la puissance de calcul actuelle du modèle de référence de temps, .i.e. le premier hôte de l'itinéraire. L'essence de la tâche de réajustement est de ramener les fragments de temps de l'hôte à la charge du travail des fragments de temps correspondant de l'hôte de référence. Nous rappelons que, la puissance de calcul d'un hôte est rendue proportionnelle à sa charge de travail en ignorant le temps des instructions noires. Ainsi, les facteurs de réajustement sont calculés sur la base des charges de travail pertinent. Concrètement, la nouvelle valeur de temps d'un fragment de temps, par exemple i, est recalculée en introduisant la formule suivante :

$$Readjusted\_Fragment_i\_time = \left\{ \frac{(referential\_host\_fragment_i\_average\_load)}{(host\_fragment_i\_average\_load)} \right\} \times host\_fragment_i\ (t_{initial} - t_{final})$$

Un hôte ordinaire .i.e. un hôte différent de l'hôte de référence, est déclaré suspect lorsqu'au moins un de ces fragments de temps réajusté est plus grand que le fragment de temps correspondant de l'hôte de référence. Cependant, l'hôte de référence est déclaré suspect quand chacun de ses fragments de temps est plus grand que chaque fragment réajusté de temps correspondant de chacun des autres hôtes. Dans le cas où au moins un hôte est suspect, le propriétaire de l'agent mobile construit ce qu'on appelle une preuve de base signée et l'envoie à l'entité de médiation. La preuve de base est composée de fragments de temps et des charges de travail correspondantes signées. Ici, il existe deux cas distincts, dépendants de l'identité de l'hôte suspect. Dans le cas où l'hôte de référence est suspect, la preuve de base doit obligatoirement inclure l'information totale de la mission .i.e. information pertinente de tous les hôtes de la journée de l'agent mobile. A l'inverse, seul l'information pertinente de l'hôte suspect et celle de l'hôte de référence sont nécessaires. La preuve de base est analysée par l'activité de médiation assignée à la tierce partie de confiance.

4.2 Protocole de Médiation

La tâche de médiation est assignée à la tierce partie de confiance. Son objectif est de vérifier si oui ou non, une revendication d'un propriétaire d'un agent est une simple allégation. Une revendication est soit une simple allégation ou au contraire le caractère

malicieux de l'hôte est déterminé. La tierce partie de confiance travaille de manière progressive, elle commence par affirmer la véracité de l'hôte de référence. Ensuite, elle vérifie la crédibilité de la revendication. Quand une confidence plus grande est gagnée, la tierce partie de confiance lance la phase ultime de l'établissement de la vérité. Les étapes conduites sont développées selon leur séquencement naturel.

4.2.1 Affirmation de la Véracité de l'Hôte de Référence

Lorsque la tierce partie de confiance reçoit un message signé, composé de la preuve de base d'un propriétaire d'un agent, elle commence par affirmer la véracité de l'hôte de référence. Nous devons rappeler que, l'exécution de l'agent chez l'hôte de référence est supposée être fiable, car elle exclut les modifications clandestines. Par conséquent, sa durée est attendue d'être minimale. Ceci représente un point crucial dans la vérification de la véracité de l'hôte de référence. A base de cette supposition, le processus de vérification de véracité commence par réajuster les fragments de temps des autres hôtes en utilisant la même formule du paragraphe précédent. Ensuite, il ramène chaque fragment de temps à un temps réel en utilisant la puissance de calcul et la charge du travail de références. Après cela, il sélectionne les durées de temps réel de tous les hôtes. Le statut de l'hôte de référence est déclaré comme celui du modèle d'exécution de référence, .i.e. de confiance si jamais la somme des temps réels minimaux est égale à la durée totale des temps réels de l'hôte de référence de l'exécution. Plus de précisions sont données par les expressions mathématiques suivantes de l'algorithme suivant où l'ordre doit être préservé :

1. For each host different of the referential host do
   For each $fragment_i$ do

$$Readjusted\_Fragment_i\_time \leftarrow \left\{ \frac{referential\_host\_fragment_i\_average\ load}{host\_fragment_i\_average\_load} \right\} \times host\_fragment_i(t_{initial} - t_{final})$$

2. For first hop Host Only
   For each $Fragment_i$_time do

$$Readjusted\_referential\_host\_fragment_i\_time \leftarrow referential\_host\_fragment_i\_time$$

For each host including the referential host do
   For each $Readjusted_fragment_i$_time do

$$Real\_fragment_i\_time \leftarrow \left\{ \frac{host\_referential\_load}{host\_fragment_i\_average\_load} \right\} \times host\_referential\_Power\_of\_Processing \times Readjusted\_Fragment_i\_time$$

3. For each $fragment_i$_time of all hosts

Select the minimal $Real\_fragment_i$ time

If ($\sum minimal\ Real\_fragment_i\_time = \sum referential\_host\_Real\_fragment_i\_time$

Then the referential host is truly the model of execution

Algorithm 4: Referential Host Veracity Checking Process

4.2.2 Crédibilité de la Revendication

Dans cette section, nous vérifions la crédibilité de la revendication en se basant sur le calcul de la formule mathématique du protocole de détection du caractère suspect [12, 13]. Quand, au moins un temps réel d'un hôte est plus grand que le temps réel du fragment correspondant de l'hôte de référence ou bien chaque temps réel de l'hôte de référence est supérieur à chaque fragment du temps réel correspondant de l'ensemble des hôtes, une plus grande confidence dans la revendication est gagnée. Autrement, la revendication ni que juste une allégation, et la tierce partie de confiance peut à priori incrémenter le degré de sévérité dans le ''log'' associé au propriétaire de l'agent mobile, lui envoyer un message d'avertissement et le sanctionner en arrivant à un certain seuil.

4.2.3 Etablissement de la Vérité

Pendant cette phase, le processus d'établissement de la vérité est activé. Il commence par demander l'information pertinente de l'hôte suspect (e.g. son prix). Juste après, il gèrera le dialogue des questions/réponses composé des propositions de la logique du premier ordre. Nous devons rappeler, que la tierce partie de confiance connait à l'avance les résultats de ses propositions. Ce dialogue passera par plusieurs sessions attachées à deux différents modes de fonctionnement. Dans le premier mode de fonctionnement, les objectifs de la gestion d'une session sont :

1. Generating suspicious host price centric questions (i.e., first order logic propositions) at random where any former questions are reinserted.
2. Detecting any suspicious host attempt to remember former questions.

3. Detecting incoherencies among sessions' responses

Le premier mode de fonctionnement restera actif tant que la tierce partie de confiance est capable de gérer efficacement les ressources nécessaires et les réponses sont compatibles (quand les réponses sont incompatibles l''hôte suspect est déclaré malicieux et le protocole de médiation s'arrête). Cependant, à cette frontière la tierce partie de confiance basculera vers le deuxième mode de fonctionnement. Ensuite, la tierce partie de confiance favorise l'initiation du processus d'établissement de la vérité au cas où elle est renseignée sur la tentative de mémoriser des questions précédentes du dialogue par l'hôte suspect. Autrement, l'hôte est déclaré honnête. Le processus de l'établissement de la vérité tente de pousser rapidement l'hôte suspect vers la frontière d'un état d'engloutissement. Avec beaucoup d'espoir, avant qu'il ne devienne défectueux, l'hôte suspect ne sera pas en mesure de donner des réponses compatibles. Pendant ce second mode de fonctionnement, les sessions de la tierce partie de confiance accomplissent les actions suivantes afin d'établir la vérité :

1. Generating suspicious host price centric questions (i.e., first order logic propositions) at random where the first mode of operating questions are reinserted.
2. Supervising responses will focus only on the average of their results. We should notice now that the Third Trusted Party cannot afford individual response supervision due to the lack of resources.
3. Terminating this second operating mode when the average of the responses results is not as expected. Thereafter, the host is declared to be malicious.

La section précédente a esquissé l'essence de notre protocole de médiation exécuté par la tierce partie de confiance. Du moment que la partie la plus cruciale a pour but de pousser progressivement l'hôte suspect à un état d'un désarroi complet, nous discutons les questions qui relèvent de l'ordre pratique lié à cet aspect particulier. Les questions d'une plus grande importance sont celles de la génération des propositions, les tailles des sessions, l'ordre aléatoire des propositions d'une session, et enfin la façon d'exprimer les propositions antérieures.

En effet, Les réponses proposées clarifient pourquoi l'hôte suspect sera inondé alors que la tierce partie de confiance ne le sera pas. En premier, nous considérons le volet des tailles des sessions. Dans son état initial le protocole peut utiliser une fonction hyper géométrique afin d'établir aléatoirement la taille de départ d'une session. Les tailles suivantes doivent augmenter pour non seulement incorporer et gérer les propositions précédentes mais ainsi que les nouvelles propositions pertinentes. Évidemment, le mécanisme de génération des propositions est d'une importance capitale. Une génération systématique des propositions nécessite des propositions pivotées par l'information de l'hôte suspect, .i.e. son prix. En pratique, un intervalle centré par le prix et incluant implicitement un ensemble de valeurs de cardinalité égale à la taille de la session est suffisant. Par conséquent, exprimer une proposition par exemple, peut être assez simple qu'une question '' is(price – a lower interval value) equal to the real value of the différence''. Le processus de génération des propositions doit véhiculer un germe permettant la détection des tentatives de mémoriser les questions antécédentes. Alors, les questions antécédentes doivent être exprimées différemment, de façon à ce que leurs mémorisations soient facilement perceptibles par la tierce partie de confiance. En réponse à cet objectif, nous utilisons un mot réservé reflétant le terme droit d'une proposition. Dans ce cas, une question antécédente est incluse dans la nouvelle proposition de façon simple à juste titre comme ''is(price – Former_Session[i].Right_Hand_term) { = , >, <} to the real value of the difference''. Concernant l'ordre aléatoire des arrangements des propositions, il est utile de signaler que des méthodes standards de la littérature informatique sont des candidats très plausibles.

5. Analyse de la Robustesse du Protocole de Détection du Caractère Suspect

Notre approche rend le traitement chez chaque hôte, proportionnel à sa charge du travail. La technique consiste à ignorer le temps d'interrogations des catalogues. Le protocole de détection du caractère suspect considère le premier hôte comme le modèle de référence de l'exécution. Par conséquent, nous ramenons les temps des traitements des autres hôtes à la vitesse de traitement correspondante de ce modèle. Non seulement, ce choix a éliminé le besoin d'utiliser des composants de gestions du référentiel général et d'estimation de temps mais aussi parce qu'il reflète une réalité élèmentaire. La réalité exprime un fait qui est ''first host has no fugitive action other than moving the roving agent to a different place''.

5.1 L'Exemple du Test

Un soin particulier est accordé à la validation de notre approche. Nous avons choisi un exemple qui nous permet d'exploiter pleinement la puissance de la solution proposée. Notre approche réduit considérablement la complexité des prérequis des deux travaux de Vigna et Esparza. Par conséquent, l'exemple de Vigna est enrichi afin d'englober les spécificités de notre approche. A titre d'exemple, au lieu de deux magasins uniquement, le nombre est élargi à trois. Ce nombre plus grand de magasins permet d'imaginer le scénario particulier où le premier hôte modifie l'itinéraire de l'agent mobile. Dans l'application des agents mobiles simple choisie, nous considérons un utilisateur au niveau du site home.sweet-home.com souhaitant acheter un film vidéo de Tarantino's Pulp Fiction movie. Par conséquent, il envoie son agent vers le site agents.virtualmall.com dédié à maintenir un catalogue des magasins électroniques. Une fois sur place, l'agent interroge le dossier du site sur les sites spécialisés dans la vente des films vidéo. Ensuite, l'agent mobile entamera sa visite aux sites fournis. Dans chaque site, l'agent contacte le service du catalogue local afin de déterminer le prix courant du film Pulp Fiction home vidéo. Lors de la collecte de tous les prix, l'agent mobile identifie à sa base le meilleur offre, et si cette offre est inférieure à une somme, .e.g. twenty dollars, l'agent mobile se place chez le site sélectionné et achète la vidéo.

5.2 Aspects Techniques

Notre agent vagabond garde dans son état les temps de traitement et les charges de travail signées des sites visités. Les frontières d'interrogation du catalogue représentent les abords des fragments de temps. Le programme faisant le test est développé en langage java et expérimenté au niveau d'une machine Pentium III 866 MHZ, sous le système Lunix Ubunto. La question traditionnelle que chacun pose est ''How fast does a Program run a Machine''?''. Dans notre discussion antérieure, nous avons admis que la réponse à cette question a une précision parfaite. En réalité, il se trouve que ce problème est vraiment complexe. Il y a plusieurs facteurs qui varient d'une exécution d'un programme à un autre. Les ordinateurs n'exécutent pas seulement un seul programme à la fois, mais continuellement basculent d'un processus à un autre, exécutant un code d'un processus avant d'exécuter un autre d'un autre processus. La planification exacte des ressources du processeur à un programme dépend de facteurs tels le nombre d'utilisateurs du système, le trafic au niveau du réseau, et le timing des opérations sur le disque. Les chaines d'accès au cache ne dépendent pas uniquement des références action-

nées par le programme dont on veut mesurer ces performances, mais de celles des autres programmes en concurrence. Notre protocole proposé résout de manière simple et efficace le problème du nombre de programmes en activité simultanés, ignore simplement les opérations du disque .i.e. les instructions noires, et neutralise les fluctuations dues au cache ce qui correspond au contexte réel des exécutions des agents mobiles en visite. Malheureusement, nous sommes incapables de mesurer la charge du travail de l'ordinateur pendant le test. Nous rappelons que la charge du système est un élément nécessaire pour la bonne application de notre protocole. Afin de surmonter ce problème de programmation, nous avons opté pour une approche statistique. Plus précisément, la méthode statistique''Hypothesis Test for the Difference between Two Means'' appelée ''two sample t_test''.

Pour les besoins de la validation, des scripts différents de programmes en java sont développés. La tâche principale des scripts est d'itérer l'exécution de l'agent mobile un nombre égal à cent. Les images différentes des programmes en java englobent le cas des exécutions intègres (.i.e. hôte honnête) et non intègre (.i.e. hôte malicieux) du modèle de l'agent mobile. Ensuite, nous avons analysé et tracé les résultats de la simulation sous forme de graphes pour une meilleure lisibilité. Comme le rang des itérations est sans importance, les fragments des temps d'exécution sont alors triés. Les graphes des différentes images du modèle de l'agent mobile sont donnés dans la section suivante.

5.3. Interprétation des Résultats de Simulation

Les résultats sous la forme de graphes décrivent les sorties de la simulation dans des situations différentes. Ces résultats sont obtenus de l'exécution des différentes images du modèle de l'agent mobile de l'exemple. Ces images différentes représentent l'exécution au niveau du modèle de référence de l'exécution, de l'hôte intègre et de l'hôte malicieux. Ci-dessous sont les graphes des fragments de temps de l'exécution des différentes images de notre agent mobile :

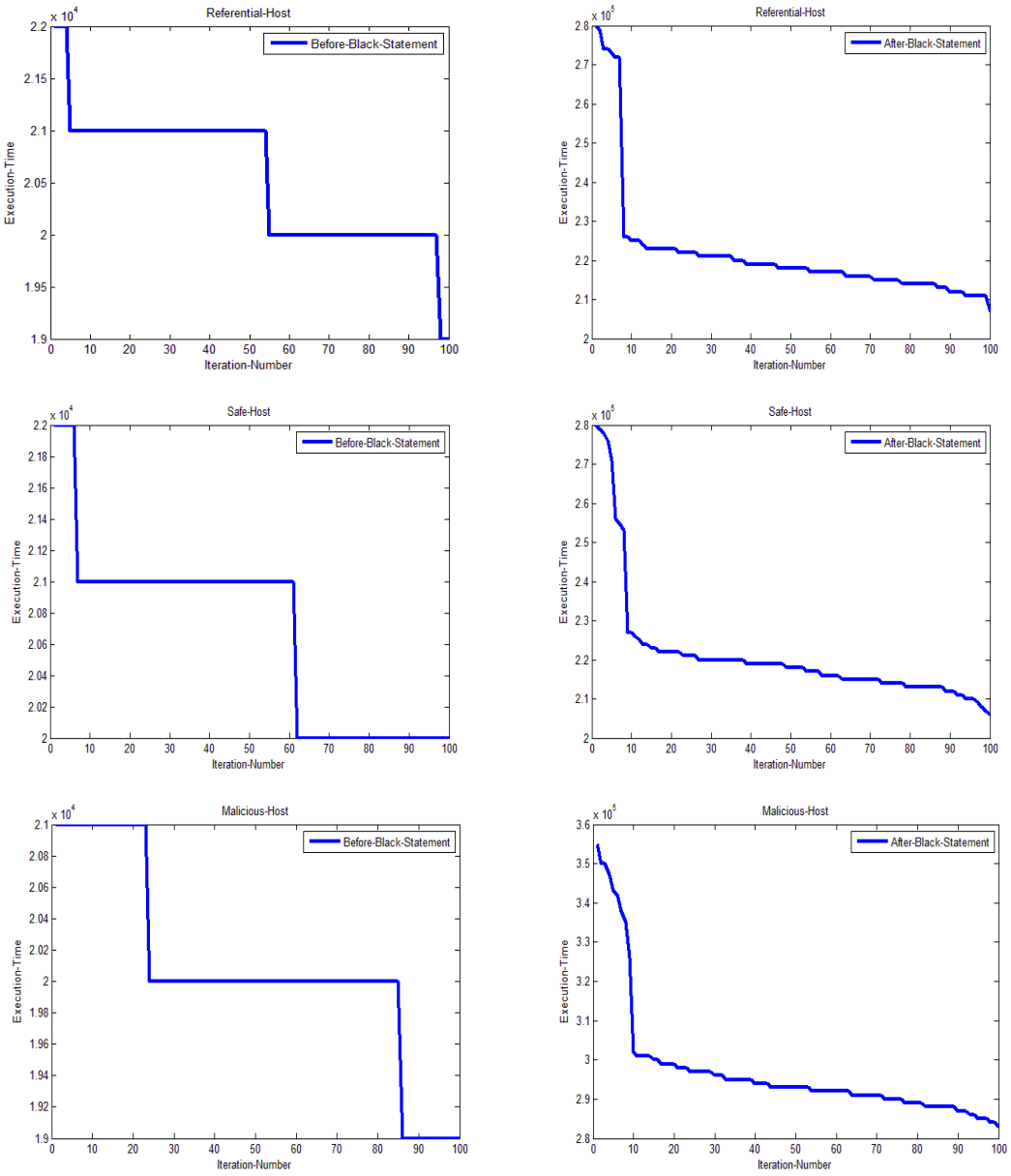

Figure 8: Hosts' Fragments of Execution Times Graphs

L'observation principale est que ces graphes ont un degré de similitude très grand vis-à-vis du premier fragment de temps, .i.e. Before Black Statement. Cependant, une grande différence est remarquée dans le deuxième fragment de temps, .i.e. After Black Statement du troisième hôte. Dès lors, on peut attester que les petites fluctuations dans les exécutions typiques de l'agent, sont dues en grande partie à un facteur endogène à l'agent vu le choix délibéré d'un Tree-Map comme structure de stockage d'information, et un facteur hétérogène originaire de l'installation de test.

Numériquement, l'information cruciale des graphes est résumée par le tableau suivant, où l'on désigne la moyenne (μ) et la déviation standard (SD) de chaque hôte de l'exemple du test :

Time In Nano Seconds	Referential_Host	Safe_Host	Malicious_Host
Before_Black_Satement	μ =20500 SD=508	μ =21000 SD=508	μ =20000 SD=508
After_Blak-Statement	μ =243500 SD=508	μ =243000 SD=508	μ =319000 SD=508

Table 4: Agent Images' Execution Times Means and Standard Deviations

En comparant les graphes, spécialement les éléments de la table ci-dessus, nous distinguons que le troisième hôte est un hôte malveillant. En effet, la moyenne de son deuxième fragment de temps d'exécution est beaucoup plus grande que la moyenne toujours du deuxième fragment de temps de l'hôte de référence de l'exécution. Ceci représente le cas où le protocole détecte le faible caractère de suspicion du troisième hôte. La troisième image du modèle de l'agent mobile de l'exemple traduit une situation anormale où l'hôte sera vainqueur car l'état de l'agent mobile a été modifié (nécessite plus de temps) afin de rendre l'offre du troisième hôte le meilleur.

Détaillons maintenant la démarche basée sur le processus de validation statistique. Notre test englobe la formulation et le test des hypothèses, des assertions qui sont falsifiables en utilisant des données observées. La **null hypothesis** est une proposition typique d'une position générale ou de défaut, telle l'absence d'une relation entre deux phénomènes mesurés. Il faut rappeler que, le protocole de détection du caractère suspect d'un hôte démarre par une vérification de la véracité du hôte de référence, .i.e. le modèle de référence de l'exécution. Ensuite, il analyse ses incohérences avec les autres hôtes de l'itinéraire.

Les données collectées pendant le déroulement du test, représentant des séries d'échantillons de même taille, montrent que les temps des exécutions génèrent la même déviation standard. Ainsi, nous suggérons de considérer que deux temps pertinents mesurés sont égaux, lorsque leur distance correspondante, .i.e. la valeur absolue de la différence est inférieure à l'unique déviation standard. Par consequent, le test statistique approprié est l'

‘’**Hypothesis Test for the Difference between Two Means**’’ appelé ‘’ **two-sample t-test**. Ce test est composé de quatre étape: (1) formuler les hypothèses, (2) formuler un plan d’analyse, (3) analyser l’échantillon des données, et (4) interpréter les résultats.

Chaque test d’hypothèses appelle l’analyste d’établir une ‘’null hypothesis’’ et son hypothèse alternative. Les hypothèses sont établies de manière à ce que toutes les deux s’excluent mutuellement. Autrement dit, si une est vraie, la deuxième est obligatoirement fausse ; et vice versa. Le tableau suivant montre trois ensembles d’hypothèse null et son hypothèse alternative. Chaque ensemble établit un fait relatif à la distance d entre les moyennes de populations à savoir μ_1 μ_2.

Set	Null hypothesis	Alternative hypothesis	Number of tails
1	$\lvert \mu_1 - \mu_2 \rvert = d$	$\mu_1 - \mu_2 \neq d$	2
2	$\mu_1 - \mu_2 \geq d$	$\mu_1 - \mu_2 < d$	1
3	$\mu_1 - \mu_2 \leq d$	$\mu_1 - \mu_2 > d$	1

Table 5: Set of Null and Alternative Hypotheses

Le premier ensemble des hypothèses (Set 1) est un exemple d’un test à deux ‘’tails’’, puisque une valeur extrême de chaque côté de la distribution de l’échantillonnage amènera le chercheur à refuser l’hypothèse null. Les deux autres ensembles des hypothèses (Sets 2 and 3) sont des tests à un ‘’tail’’, puisque une valeur extrême au niveau d’un seul côté de la distribution de l’échantillonnage amènera un chercheur à refuser l’hypothèse ‘’null’’.

Le plan d’analyse décrit la manière d’utiliser les données de l’échantillon afin d’accepter ou de refuser l’hypothèse ‘’null’’. Il doit spécifier les éléments suivants :

- ✓ Choosing the significance level.
- ✓ Using the two-sample t-test to determine whether the difference between means found in the sample is significantly different from the hypothesized difference between means.

L'analyse de l'échantillon de données, détermine l'erreur standard, le degré de liberté, test statistique, et la P-value associée au test statistique.

- ✓ Computing the standard error (SE) of the sampling distribution by:

  $SE = sqrt[(s_1^2/n_1) + (s_2^2/n_2)]$ where s_1 is the standard deviation of sample1, s_2 is the standard deviation of sample 2, n_1 is the size of sample 1, and n_2 is the size of sample 2.

- ✓ Calculating the degrees of freedom (DF) is:

  $DF = (s_1^2/n_1 + s_2^2/n_2)^2 / \{[(s_1^2 / n_1)^2 / (n_1 - 1)] + [(s_2^2 / n_2)^2 / (n_2 - 1)]\}$

- ✓ Defining the test statistic as t-score (t) defined by the following equation.

  $t = [| (x_1 - x_2) |- d] / SE$ **where** x_1 is the mean of sample 1, x_2 is the mean of sample 2, d is the hypothesized difference between population means, and SE is the standard error.

- ✓ Establishing by calculator the P-value which is the probability of observing a sample statistic as extreme as the test statistic. Since the test statistic is a t-score, use the t Distribution Calculator to assess the probability associated with the t-score, having the degrees of freedom computed above.

L'interprétation des résultats conclut que si les conclusions de l'échantillonnage sont improbables, alors étant donnée l'hypothèse ''null'', cette hypothèse est refusée. Ceci implique la comparaison de la P-value au degré de signifiance, et refuser l'hypothèse null quand le P-value est inférieure au degré de signifiance.

De retour à notre test réel, nous remarquons que SE et DF sont les mêmes dans tous nos échantillons car nos échantillons ont la déviation standard qui est 508. Par conséquent, SE et DF ne sont calculés qu'une seule fois, leurs valeurs respectives sont 71.842049 et 198. La table suivante décrit la table des hypothèses concernant la vérification de la véracité de l'hôte de référence de l'exécution :

HostToHost	Before_Black_Statement	After_Black_Statement
Referential_Host To Safe_Host	$\mid \mu_{\text{Referential}} - \mu_{\text{Safe}} \mid = ?500 \leq 508$ meaning that they are tie	$\mid \mu_{\text{Referential}} - \mu_{\text{Safe}} \mid = ?500 \leq 508$ meaning that they are tie
Referential_Host To Malicious_Host	$\mid \mu_{\text{Referential}} - \mu_{\text{Malicious}} \mid = ?500 \leq 508$ meaning that they are tie	$\mid \mu_{\text{Referential}} - \mu_{\text{Malicious}} \mid = ?500 \leq 508$ meaning that they are tie

Table 6: Working Example Null Hypothesis

En substituant les valeurs de l'exemple du test, les valeurs du t-scores et du P-values correspondantes sont résumées dans le tableau ci-après :

ReferentialHost To aHost	Before_Black_Staement		After_Black_Statement	
	t-score	P-Value	t-score	P-Value
Referential_Host To Safe_Host	0	1	0	1
Referential_Host To Malicious_Host	0	1	1043.9569	0

Table 7: Null Hypothesis t-scores and P-Value

A l'exception de l'hypothèse ''null'' de ''After_Black_Statement'' où la P-value est égale à zéro, l'hôte de référentiel de l'exécution est assuré d'un temps d'exécution minimal partiel (la probabilité égale à 1 parle d'une certitude). Par conséquent, si nous arrivons à prouver que $\mu_{\text{Malicious}} - \mu_{\text{Referential}} > 508$ soit disant 509, nous pouvons affirmer formellement que l'hôte de référentiel de l'exécution est réellement le modèle de référence de l'exécution et le troisième hôte a exécuté des actes malicieux. Comme c'est un test de un ''tail'', où le t-score est égal à ((75500–509)/71.842049)= 1043.8316, le calculateur nous informe que : P(t

<= 1043.8316) = 0.0. Par conséquent, la P-value est : p(t>= 1043.8316) = 1 – P(t<= 1043.8316) = 1.

Le fait que la simulation est élaborée sur le même ordinateur de facto élimine le besoin d'effectuer le calcul des fragments de temps réels. Par conséquent, notre conclusion précédente .i.e. le troisième hôte s'est comporté de manière malicieuse reste toujours valide. On doit tout de même signaler que, l'exemple présenté qui relate une situation d'actes malicieux n'est pas du tout spécifique mais plutôt, exprime une situation générique où des actes malicieux sont exécutés. Par conséquent, nous sommes en mesure de conclure que la simulation a conforté le protocole dans ces objectifs tracés et que sa robustesse est validée.

6. Conclusion

Ce chapitre a motivé, exploré, et définit un protocole original dans sa vision de traitement du problème complexe des hôtes malicieux. Au centre de sa démarche, il y a deux idées principales. Les origines de ces deux idées sont les travaux de Vigna et d'Esparza. En l'occurrence la notion des instructions noires des traces cryptographiques de Vigna et le principe de la limitation du temps d'exécution de l'agent mobile chez les hôtes d'Esparza. L'originalité du protocole réside dans la manière nouvelle dont il procède afin de détecter les actes malicieux des hôtes. Le protocole développé est robuste et a un coût faible, .i.e. simple à implémenter. Par sa manière progressive d'agir, le protocole entame sa tâche par détecter le faible caractère de suspicion des hôtes. L'activité de l'établissement de la vérité finit le travail en infirmant ou en confirmant les comportements malicieux. La procédure de la collecte de l'information pertinente de décidabilité du caractère suspect est simple et facile à réaliser. Elle permet une discrimination préalable des exécutions intègres de celles malveillantes. La tâche complexe de l'établissement de la vérité n'est entamée qu'en cas d'une probabilité assez forte de comportement vraiment malveillant. Dans le cas où le caractère malicieux viendrait d'être confirmé, un protocole de révocation des hôtes est appliqué. Par conséquent, la tierce partie de confiance dont la tâche est plus élaborée ici, contribue efficacement au procédé d'assainissement global de l'Internet.

Conclusion Générale

L'espoir né de l'apparition des agents mobiles en 1994 s'est vite confronté aux problèmes sérieux de sécurité. Les attentes engendrées par les agents mobiles sont grandes et multiples. On note particulièrement, la façon dont les applications distribuées sont développées. Les technologies, architectures, et méthodologie traditionnellement utilisées dans le développement des applications distribuées souffrent d'une variété de restrictions et obstacles au niveau d'un environnement de la taille d'Internet. La mobilité du code est considérée par beaucoup de chercheurs comme l'approche la plus prometteuse aux problèmes : (1) de souplesse de configuration ''configurability''. (2) de souplesse de croissance ''scalability''. (3) de personnalisation ''customizability''. Cependant, les agents mobiles n'ont pas eu l'essor qui leur a été prédestiné, d'emploi et de déploiement massifs. Le développement à base d'agents mobiles requiert une architecture sous-jacente sécuritaire; particulièrement dans le cas des applications du e-commerce. Deux problèmes majeurs sont liés à l'aspect sécuritaire des agents mobiles : (1) La protection des hôtes d'attaques des agents. (2) La protection des agents mobiles d'attaques des hôtes dits malicieux.

Le problème de la protection des agents mobiles contre les attaques d'hôtes malicieux est beaucoup plus difficile à résoudre [9]. Plusieurs solutions ont été proposées sans qu'aucune ne soit pleinement satisfaisante. Ces solutions varient considérablement, parmi lesquelles, on n'y retrouve des solutions d'un niveau purement matériel jusqu'au niveau purement logiciel [44]. Sans échapper à la tradition, les deux grandes orientations de ces solutions sont : la prévention et la détection.

Le problème délicat des hôtes malicieux dans Internet rend l'utilisation et le déploiement des agents mobiles une opération hasardeuse. L'objectif principal de notre thèse était de proposer un procédé d'assainir Internet de ces Hôtes. La situation chaotique que provoquerait la présence de tels hôtes est sans dire inadmissible. La thèse propose une approche uniforme de lutte contre le phénomène des hôtes malveillants. L'approche préconise l'utilisation d'une entité de médiation dans deux solutions. La première solution est une extension de la technique préventive développée par Minsky. Par contre, la deuxième solution est une toute nouvelle technique de détection.

L'un des objectifs de la thèse est de montrer la latitude de justifier la standardisation de la résolution du problème des hôtes malicieux par la médiation. Un deuxième est de montrer que la médiation contribue aussi à simplifier la complexité inhérente aux techniques de protection des agents mobiles. L'intégration de la médiation à la technique préventive de Minsky a un coût faible [44]. C'est un travail d'effet de bord car à l'origine le protocole garantit l'intégrité de l'exécution des agents mobiles. Concernant, la solution de détection développée [12, 13], il est impératif de constater que la finalité d'une approche de détection est de recourir toujours à un recouvrement. Dans ces techniques de détection, une tierce partie de confiance .i.e. un arbitre est toujours impliqué pour proclamer le verdict. Construire sur cette base une médiation n'est au fait qu'un prolongement logique aux compétences de l'arbitre. Mandater l'arbitre d'une prérogative, qui va au-delà d'une simple vérification de la véracité du caractère malveillant des hôtes, a considérablement réduit la complexité de la technique de détection développée.

Pour développer le procédé de médiation chez les techniques préventives, le chapitre 3 a défini les critères qui ont motivé le choix du protocole de traitement parallèle. La procédure de validation du protocole proposé est conduite dans un cadre formel. Le cadre formel des systèmes OO-Action à mobilité est étendu par deux nouvelles actions de clonage et de déplacement d'un groupe d'agents à des localisations physiques différentes. Un point important partagé entre les deux actions est qu'elles sont atomiques. Cette propriété est d'autant plus importante, qu'elle contribue à déceler quelques types de déclarations mensongères des hôtes. L'extension du protocole de Minsky dont la médiation en fait partie, distingue à priori un hôte de confiance. Un hôte de confiance au sens de l'extension est un hôte qui a délivré la preuve de sa disposition à exécuter l'agent en parfaite accord avec sa politique publiée. La preuve est un certificat octroyé auprès d'une autorité compétente qui vérifie périodiquement la validité des politiques publiées des hôtes. La collecte des preuves de confiance, surnommées évidences de coopération, est régie par un protocole probabiliste de non répudiation. Le médiateur de l'extension est capable de : (1) déterminer les hôtes malicieux et d'en révoquer probablement ceux qui sont récidivistes. (2) de discerner les comportements bizarres des hôtes de confiance et de faire en sorte qu'ils soient disséminés.

En conformité avec l'uniformité de la médiation, un nouveau protocole de détection des hôtes malicieux est développé en chapitre 5. Ce protocole utilise de manière originale les deux concepts à savoir les instructions noires de Vigna et la limitation du temps d'exécution

d'Esparza. La première partie du protocole contourne quelques difficultés de la technique d'Esparza en ignorant le temps d'exécution des instructions noires. Par conséquent, le temps d'exécution d'un agent mobile est proportionnel à la charge du travail d'un hôte. Au second plan, cette partie prévoit d'attribuer au premier hôte de l'itinéraire le statut d'un hôte du modèle de l'exécution de référence. La véracité de ce statut est vérifiée en fin de séjour de l'agent mobile de la part de son propriétaire. Un caractère plus faible qualifiant un ou plusieurs hôtes est établi dans une deuxième phase. Cette phase en est même capable d'identifier aussi le caractère suspect du modèle de l'exécution de référence. Dans sa deuxième partie, le protocole sollicite l'autorité compétente .i.e. le médiateur pour infirmer ou confirmer le caractère plus fort d'être malicieux d'un ou de plusieurs hôtes. Le médiateur par un jeu de questions réponses pertinentes tentera de déceler les incompatibilités parmi les réponses. Le travail du médiateur est composé de deux phases. Dans chacune, une série de sessions de questions réponses est entreprise. La taille des sessions dépend d'une fonction hyper géométrique. La deuxième phase du travail du médiateur est activée lorsque les demandes en ressources du médiateur avoisinent un certain seuil. Auquel cas, le médiateur bascule vers le calcul des moyennes des résultats de réponses.

Dans l'objectif de valider la première partie de ce protocole, un test à base de scripts java décrivant les situations d'une exécution intègre, et d'une autre malveillante est opéré sur un système Lunix Ubunto. Les résultats de la simulation traités et validés par la méthode statistique Two Sample t Test confirment absolument la robustesse de cette partie du protocole liée au caractère suspect d'un ou de plusieurs hôtes.

Tout travail de recherche nécessite certainement des améliorations, des extensions, des validations des parties qui sont au stade de la spécification ou de la conception. Pour le travail de notre thèse, nous pensons qu'une généralisation du protocole de Minsky étendu, au cas d'un traitement parallèle à plusieurs phases ''Stages'' est le point de départ à toute consolidation future aux termes acclamés par notre approche. Même si ce point semble à portée de mains, il sera utile pour dériver formellement via le calcul de raffinement une version en langage de programmation prête à l'analyse expérimentale à partir de la spécification de départ. Une autre perspective de cette thèse serait celle d'expérimenter réellement la deuxième partie du nouveau protocole de détection des actes malicieux. De notre avis, cette tâche est délicate et nécessite un effort de recherche conséquent

supplémentaire. Tout semble à croire que, ramener des expressions syntaxiques libres et distincts vers une même sémantique via l'application des ontologies, permettrait de :

1. Rendre le problème de vérification des incohérences parmi les éléments des sessions lors d'une médiation, un problème décidable,
2. Libérer les hôtes et les TTPs de s'exprimer dans un même langage, surtout, si elles appartiennent à des autorités distincts de l'Internet.

Références Bibliographiques

[1] J.E. White. Telescript Technology : the foundation for Electronic Market-Place. Technical Report, General Magic, Inc. 1994.

[2] R. H. Guttman, A. G. Moukas, and P. Maes. Agents as Meditors in Electronic Commerce. International Journal of Electronic Markets; 8(1): 22-27, 1998.

[3] S. Robles. Mobiles Agents Systems and Trust, a Combined View toward Secure Sea-of-Data Applications. PHD thesis. Universitat Autonoma de Barcelona, 2002.

[4] Dja Milojicic. Mobile Agent Applications. IEEE Concurrency, 07(3) : 80-90, 1999.

[5] Carles Garrigus Olivella. Contribution to Mobile Agent Protection From Malicious Hosts. PHD thesis. Universitat Autonoma de Barcelona, 2008.

[6] J. Amettler, S. Robles, and J. A. Ortega. Self Protected Mobile Agents. In Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagents Systems. (AAMAS' 04). Pages 362-367. IEEE Computer Society. 2004.

[7] J. Zhou, J. A. Onieva, and, J. Lopez. Analysis of free roaming agent result-truncation defense Scheme. In Proceedings of the IEEE Int. Conf. On e-Commerce Technology (Cec' 04), Pages 221-226. IEEE Computer Society, 2004.

[8] F. L. Bellifemine, G. Caire, and D. Greenwood. Developing Multi-Agent Systems with JADE. John Wiley & Sons, 2007.

[9] R. S. Gray, D, Kotz, G. Cybenca and D. Rus. D'Agents: Security in Multiple-Language, Mobile-Agent System. In Mobile Agents and Security, Volume 14/9 of Lecture Notes in Computer Science, pages 154-187. Springer Verlag, 1998.

[10] N. Borselius. Mobiles Agent Security. Electronic Communication Engineering Journal, 14(5) : 211-218, 2002.

[11] Lora L. Kassab and Jeffry Voas. ''Agents Trustworthiness''. Ecoop Workshop, 1998:300.

[12] M. Chihoub ‘’Malicious Hosts Detection Through Truth Powered Approach’’, In Proceeding of Third International annual Workshop on Digital Forensics and Incident Analysis, IEEE Computer Society, Malaga, Spain 9 October 2008.

[13] M. Chihoub, R Maamri, Design and Efficiency Analysis of a New Protocol for Malicious Hosts Detection. International Journal of Smart Home Vol.6 N01, January, 2012.

[14] Giovanni Vigna. ‘’Protecting Mobile Agents Through Tracing’’. In Proceeding of the 3rd Ecoop Workshop on Mobile Objects Systems, Jyvalskyla, Finland 1997.

[15] F.B. Schneider, "Towards Fault-tolerant and Secure Agentry", in Distributed Algorithms, Mavronicolas et Tsigas (Ed.)**,** Springer-Verlag, Berlin, Germany 1997, pp. 1-15.

[16] Y. Minsky, and al, Cryptographic Support for Fault-Tolerant Distributed Computing. July 5, 1996.

[17] O. Esparza, M. Soriano, J.L. Munoz, and J. Forné. ‘’A Protocol for Detecting Malicious Hosts Based on Limiting the Execution Time of Mobile Agents’’. In IEEE Symposuim on Computers and Communications-ISCC’2003, 2003.

[18] S. Goutet, "Mobile Agents Technologies : Implementation and Evaluation", Technical Report, Ericsson, Avril 2000.

[19] B, Lenou, "Services sur réseaux mobiles : Architectures et maintenance ", Mémoire de Maîtrise, École polytechnique de Montréal, Canada, Décembre 2*000***.**

[20] D**.** Chess, C. Harrisson, A**.** Kershenbaum, Mobile Agents : Are they a good idea, IBM Research Report RC 19887 (88465), December 1994.

[21] D. Chess, B. Grosof, C. Harisson, D. Levine, C. Parris, Itinerant Agents for Mobile Computing, IBM Research Report, 1995.

[22] W. A. Jansen, T. Kaygiannis, "Mobile Agent Security", NIST Special Publication 800- 19, 1999.

[23] M. S. Greenberg, J. C. Byington, T. Holding, D. G. Harper, " Mobile Agents and Securify", IEEE Communications Magazine, juillet 1998, pp. 76-85.

[24] W. Farmer, J. Guttman, and V. Swamp, "Security for Mobile Agents : Issues and Requirements", In Proceedings of the 19th National Information Systems Security Conference, Baltimore, Maryland., October 1996, pp. 591-597.

[25] J. J. Ordille, "When agents roam, who can trust ?', Proceedings of the Firs~ Conference on Emerging Technologies and Applications in Communications, Portland, Oregon, May 1996.

[26] G.C. Necula, P. Lee, " Safe, Untrusted Agents Using Proof-Carrying Code", Mobile Agents and Security, G. Vigna (Ed.), Springer-Verlag, Berlin, Germany, 1998. pp. 61-91.

[27] J. Gosling, H. Mc Gilton, 'The **Java** Language Environnement : A White Paper", May 1996.

[28] V. Roth, "Mutual Protection of Co-operating Agents", in Secure Intemet Programming, Vitek et Jensen (Ed.), S pringer-Verlag, Berlin, Germany, 1998, pp 26-37.

[29] V. Roth, "Secure Recording of Itineraries Through Cooperating Agents", Proceedings of the ECOOP Workshop on Distributed Object Security and 4" Workshop on Mobile Object Systems : Secure Internet Mobile Computations, pp 147-154, INNA,
France, 1998.

[30] F. Hohl, " A protocot to Detect Malicious Host Attacks by Using Reference States", Technicd Report Nr., Faculty of Informatics, University of Stuttgart, Germany, August 1999.

[31] F. Hohl, "A Frarnework to Protect Mobile Agents by Using References States", Proceedings of ICDCS 2000,2000.

[32] T. Sander, C. F. Tschudin, "'Protecting Mobile Agents Against Malicious Host", in Mobile Agents and Security, G. Vigna (Ed.), Springer-Verlag, Berlin, Germany 1998, pp 44-60.

[33] F. Hohl "Time Limited Blackbox Security : Protecting Mobile Agents from Malicious Hosts", in Mobile Agents and Security, G. Vigna (Ed.), Springer-Verlag, Berlin, Germany 1998, pp. 92- 1 13.

[34] J. Riordan, B. Schneier, "Environmental Key Generation Towards Clueless Agents", in Mobile Agents and Security, G. Vigna (Ed.)**,** Springer-Verlag, Berlin, Germany 1998, pp 15-24.

[35] P. Lee, G. Necula, "Research on Proof-Carrying Code for Mobile-Code Security TT. DARPA Workshop on Foundations for Secure Mobile Code Workshop, March 1997.

[36] A. Young, M. Yung, "Slinding Encryption : A Cryptographic Tool for Mobile Agents", Proceedings *of the 4'" intentational Workshop on Fast oftware* Encryption, FSE'97, Biham (Ed.), Springer-Vedag, Berlin, Germany. january 1997, pp 230-241.

[37] W. G. Wilhem, S. Staamann, L. Bunyan, " Introducing Third Parties to the Mobile Agent Pradigm", in Secure Internet Programming, Vitek et Jensen (Eds.), Springer-Verlag, Berlin, Germany, 1998, pp 469-492.

[38] R. A. Bazzi, "Code Hiding for Mobile Agents Security", Technical Report TR 11 12998, Department of Computer Science and Engineering, Arizona State University, November 1998.

[39] T. Sander and Christian F.T Tshudan. ‘’On Software Protection via Function Hiding’’. submitted to the 2 International Workshop on Information Hiding.

[40] Uwe G. Wilhem, Sebastian M. Staamann, and Levente Buttyan. ‘’ A Pessimistic Approach to Trust in Mobile Agent Platforms’’. IEEE Internet Computing September-October 2000. http://computer.org/internet/

[41] S. Dobrev and Al. ‘’Mobile Search for a Black Hole in an Anonymous Ring’’. Proceeding of the 15th International Conference on Distributed Computing (London, UK, 2001).

[42] Robert S. Gray, David Kotz, and Ronald A. Peterson, jr. ‘’Mobile-Agents Versus Client/Server Performance : Scalability in Information-Retrieval Task. Technical Report TR 2001-386, Junuary 30, 2001. Darmouth College Computer Science.

[43] C. Meadows, "Detecting Attacks on Mobile Agents", DARPA Workshop on Foundations for Secure Mobile Code Workshop, March 1997.

[44] M. Chihoub Mediating Hosts Malicious Character. EC2ND 2005. Proceedings of the First European Conference on Computer Network Defence. School of Computing, University of Glamorgan, Wales, UK.

[45] Lugia Petre and Kaisa Sere. Coordination Among Mobile Objets. Turku Centre for Computer Science, TUCS Technical Report Report N0 219, November 1998.

[**46**] Stan Franklin et Al. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, architectures, and Languages, Springer-Verlag, 1996.

[**47**] Wooldridge, Michael and Nikolas R. Jennings(1995).''Agents Theories, Architectures, and Languages : a Survey'' in Wooldridge and Jennings Eds, Intelligent Agents, Berlin : Springer-<verlag, 1-22.

[**48**] Russel, Stuart J : and Peter Norvig (1995), Artificiel Intelligence :A Modern Approach, Englewood Cliffs, NJ : Prentice Hall.

[**49**] Carl Hewitt, Erick Meijer, and Clemnts Szyperski '' The Actor Model''. Microsoft, April 9, 2012.

[**50**] Alfono Fuggetta, Gian Pietro Picco, and Giovani Vigna. ''Understanding Code Mobility''. IEEE TRANSACTION ON SOFTWARE ENGINEERING, VOL.24, No.5, May 1998.

[**51**] Michael S. Geenberg and Jennifer C. Byington, theophany Holding, David G. Harper. ''Mobile Agents and Security''. IEEE Communication Magazine. July 1998.

[**52**] E.W. Dijkstra. A Discipline of Programming. Prentice-Hall International, 1976.

[**53**] R.J.R. Back and K. Sere. Stepwise refinement of action Systems. Structured Programming 12: 17_30, 1991.

[**54**] R.J.R. Back and J.V. Wright. Refinement Calculus: A Systematic Introduction. Graduate Texts in Computer Science, Springer-Verlag (1998)

[**55**] Bennet S. Yee, A Sanctuary for Mobile Agents, Technical Report CS97-537, University of California in San Diego, Avril 1997.

[**56**] P. Ciancarini and C. Hankin, editors. Coordination'96: Coordination Languages and Models. Volume 1061 of Lecture Notes in Computer Science. Springer-Verlag, 1996.

[**57**] D. Garlan and D. le Métayer, editors. Coordination '97: Coordination Languages and Models, Volume 1282 of Lecture Notes in Computer Science, Springer-Verlag, 1997.

[**58**] N. Carriero and D. Gelernter. Coordination Languages and their Significance. Communication of the ACM, 35(2):97-107, February 1992.

[**59**] F. Arbab. The IWIM Model for Coordination of Concurrent Activities. CiteSeer.

[**60**] J. Zhou and Gollman. An Efficient Non-Repudiation Protocol. In Proceeding of the 10th Computer Security Foundations Workshop, pages 126-132. IEEE Computer Society Press, June 1997.

[**61**] Y. Han. Investigation of Non-Repudiation Protocols. In ACISP: Information Conference, Volume 1172 of Lecture Notes in Computer Science, pages 38-47 Springer-Verlag, 1996.

[**62**] P. Syverson. Weakly Secret bit Commitment: Application to Lotteris and Fair Exchange. In Proceeding of the 1998 IEEE Computer Security Foundations Workshop(CSFW11), June 1998.

[**63**] Olivier Markowitch, Yves Roggman. Probabilistic Non-Repudiation without Trusted Party. Second Conference on Security in Communication Networks(SCN9^0), Amalfi, Italy, September 1999.

[**64**] Mark Utting. PHD Thesis, University of New South Wales, Kensington, Australia: An Object-Oriented Refinement Calculus with Modular Reasoning, April 8, 1992.

[**65**] A. Barak and Y. Kornatzky. ''Design Principles of Operating Systems for Large Scale Multicomputers''. IBM Research Division, T.J. Watson Research Center. New York. RC13220(#59114) (1987).