Rapport INRIA 1994 — Programme 1

Système réparti tolérant les fautes et les intrusions

PROJET SATURNE

3 mai 1995

PROJET SATURNE

Système réparti tolérant les fautes et les intrusions

Localisation : *Toulouse*

Mots-clés : sécurité informatique (1), sûreté de fonctionnement (1), système réparti (1), tolérance aux fautes (1).

1 Composition de l'équipe

Responsable scientifique

Yves Deswarte, DR Inria

Personnel Inria

Jean-Charles Fabre, CR

Personnel LAAS-CNRS[1]

Mohamed Kaâniche, CR CNRS
David Powell, DR CNRS

Chercheurs doctorants

Marc Dacier, boursier CaberNet
Vincent Nicomette, boursier $\mathcal{LIS}$
Rodolphe Ortalo, boursier CNRS-UAP
Tanguy Pérennou, boursier MESR

Autres personnels

Corinne Arnaud, stagiaire DESS
Françoise Cabrolié, stagiaire CNAM

[1] Temps partiel

François Peigne, stagiaire DEA
Frédéric Salles, stagiaire DEA
Eric Totel, stagiaire DEA

2 Présentation du projet

Le projet SATURNE est un projet commun INRIA-LAAS, mené au LAAS du CNRS au sein du groupe de recherche *Tolérance aux fautes et Sûreté de Fonctionnement informatique (TSF)* dirigé par Jean-Claude Laprie. Lors de sa création, le projet SATURNE a permis d'accroître et d'étendre les activités du groupe TSF et a bénéficié en contre partie des compétences de ce groupe. De ce fait, les activités du projet SATURNE sont très imbriquées dans celles du groupe de recherche TSF.

L'objectif du projet SATURNE est l'approche globale de la tolérance aux fautes accidentelles et aux fautes d'interaction intentionnelles (intrusions) dans les systèmes répartis. Cette approche permet d'appréhender de façon unifiée les problèmes de fiabilité et de sécurité-confidentialité.

Dans le cadre du projet SATURNE, une méthode originale de tolérance aux fautes préservant la confidentialité des informations a été développée : la fragmentation-redondance-dissémination [8] [13].

La fragmentation-redondance-dissémination (FRD) consiste à découper un ensemble d'informations en fragments, de telle sorte que les informations contenues dans chaque fragment soient non significatives, à ajouter de la redondance à ces fragments pour permettre de tolérer la destruction ou la modification d'une partie des fragments, puis à disséminer ces fragments sur différents sites du réseau. Cette technique est abordée dans une optique de complémentarité avec des techniques existantes de la sécurité, comme la protection et la cryptographie par exemple.

Cette technique de FRD a d'abord été appliquée dans le cadre du projet ESPRIT DELTA-4 à un service réparti d'archivage de fichiers, puis à la gestion de la sécurité dans un système réparti. Aujourd'hui ces travaux se développent pour améliorer la fiabilité des traitements portant sur des données confidentielles (cf. §3.1).

En parallèle, une étude est menée sur la protection pour des objets à faible granularité et sur des noyaux de sécurité-innocuité et de sécurité-confidentialité (cf. §3.2). Par ailleurs, les travaux dans le domaine de l'évaluation quantitative de la sécurité se sont poursuivis (cf. §3.3).

3 Actions de recherche

3.1 Traitements fiables de données confidentielles

Participants : Jean-Charles Fabre, Yves Deswarte, Vincent Nicomette, Tanguy Pérennou

3.1.1 Contexte général

La FRD est une approche unifiée permettant de rendre tolérante aux fautes (accidentelles et intentionnelles) une application quelconque traitant de l'information confidentielle [13]. Cette technique a été utilisée dans un premier temps pour développer un système de gestion de fichiers persistants et un serveur réparti de gestion de la sécurité ; d'abord mis en œuvre sur le système expérimental DELTA-4, ces services sont maintenant portés sur un réseau de stations Sun sous Unix.

Les travaux menés jusqu'à présent ont permis de définir une méthode générale de conception qui consiste à transformer l'application en un ensemble d'objets, en minimisant ceux qui traitent de l'information confidentielle. Les objets confidentiels sont placés sur des sites dignes de confiance, les objets non confidentiels produits par cette démarche pouvant être traités de façon répliquée sur des sites géographiquement répartis non dignes de confiance [9].

Du point de vue de la mise en œuvre, une approche basée sur la propriété de *réflexivité* de certains langages objets a été étudiée et des expérimentations ont été menées en utilisant le langage Open-C++ [14].

Les travaux actuels consistent à élargir l'utilisation des méta-objets pour la tolérance aux fautes, mais aussi pour la sécurité. Ils conduiront à définir une hiérarchie de classes de méta-niveau permettant de définir les méta-objets qui conviennent pour une application donnée compte tenu de son environnement. La mise en œuvre de ces méta-objets sur micro-noyau est à l'étude.

3.1.2 Traitement fragmenté - disséminé orienté-objets

La FRD orientée-objets consiste à transformer une application en considérant la confidentialité des données manipulées par chacun des objets qui la compose. Idéalement, son objectif est de concevoir l'application sous la forme d'une collection d'objets non confidentiels (objets-fragments). Les objets-fragments ainsi identifiés seront disséminés de

telle sorte que chaque groupe d'objets localisés sur un même site ne constitue pas une information confidentielle.

Les travaux relatifs à la FRD orientée-objets sont effectués dans le cadre du projet recherche de base esprit PDCS-2 en collaboration avec Brian Randell et son équipe à l'Université de Newcastle-upon-Tyne (cf. §5.2).

Fragmentation orientée-objets

La conception orientée-objets d'une application permet d'effectuer la fragmentation en s'appuyant sur la structuration introduite par le concepteur. A un niveau d'abstraction donné, la démarche générale consiste à analyser les objets de l'application qui manipulent de l'information confidentielle. Cette analyse conduit à substituer aux objets confidentiels un ensemble d'objets, dont certains ne sont pas confidentiels. Cette démarche récursive s'arrête dès qu'il n'existe plus d'objet confidentiel auquel puisse être substitué un ensemble d'objets non confidentiels, soit à cause du type de l'objet, soit pour des raisons de performance.

Les objets issus de la conception qui manipulent cependant de l'information confidentielle devront être implantés de manière sûre au sens de la sécurité, soit en les plaçant dans une zone de confiance du système, soit en utilisant des techniques complémentaires de protection ou de chiffrement.

Redondance

La fragmentation peut, par elle-même, introduire de la redondance ; c'est le cas des techniques de schémas à seuil par exemple. Dans le cas contraire, différentes techniques de traitement d'erreur sont envisageables soit lors de la conception, soit lors de l'installation (configuration) de l'application.

Notre approche actuelle consiste à définir la technique utilisée non pas à la configuration, mais lors de la conception en utilisant certaines propriétés de la programmation objet. La propriété de réflexivité apparaît actuellement comme la plus prometteuse (cf. §3.1.3.) car elle permet de gérer la redondance en manipulant les abstractions internes du langage.

Dissémination orientée-objets

La dissémination consiste, à l'issue de la fragmentation, à créer des instances autonomes des objets sur les différents sites d'un réseau. Les deux problèmes essentiels que pose la dissémination sont les suivants :

- Gestion des regroupements d'objets

  L'expression formelle des liens entre objets issus de la phase de fragmentation (contraintes de confidentialité) permet d'éviter de grouper des objets reconstituant une information confidentielle. Le formalisme actuel repose sur une représentation ensembliste de ces liens.

- Création d'objets autonomes

  Chaque objet issu de la conception doit être instancié à distance sur un site du réseau. La difficulté de réalisation de cette opération dépend de l'environnement opérationnel réparti sous-jacent. Un support d'exécution réparti orienté-objet est bien sûr la solution idéale. Notre approche actuelle est abordée au §3.1.3.

Expérimentation

Cette démarche a été expérimentée dans le cadre de SATURNE. L'application choisie, un agenda électronique réparti, a été développée et implémentée sur le système expérimental du projet DELTA-4 ainsi que sur un réseau de stations Sun sous Unix. Ces deux démonstrations sont actuellement opérationnelles.

3.1.3 Implémentation de caractéristiques non fonctionnelles

Les travaux que nous effectuons ont pour buts l'étude et l'expérimentation de l'intégration de caractéristiques non fonctionnelles telles que la tolérance aux fautes et la sécurité dans les applications réparties. Le modèle orienté-objet semble particulièrement intéressant tant du point de vue de la description fonctionnelle de l'application que du point de vue de la mise en œuvre de caractéristiques non fonctionnelles.

Une première tentative a consisté à utiliser l'héritage pour la mise en œuvre de caractéristiques non fonctionnelles. Cependant l'héritage ne permet pas de mêler les aspects fonctionnels et non fonctionnels de manière satisfaisante. C'est la propriété de "*réflexivité*" de certains langages

objets et en particulier sa mise en œuvre sous la forme de "*protocoles à méta-objets*" qui permet de compléter de façon satisfaisante la réponse au problème.

En effet, cette propriété (initialement non spécifique du modèle objet) permet de manipuler et d'agir sur le fonctionnement interne du langage. Si l'on considère par exemple un langage interprété, cette propriété permet d'agir, dans le langage lui-même, sur le fonctionnement de l'interpréteur, non seulement du point de vue comportemental, mais également du point de vue représentation des données. Elle permet ainsi de prendre en compte de façon transparente le traitement d'erreur dans le comportement interne de l'objet et de lui associer une, voire plusieurs caractéristiques non fonctionnelles. La notion de méta-objet est relative à un méta-niveau de programmation dans lequel le fonctionnement interne des objets peut être redéfini ou adapté à une caractéristique non fonctionnelle, la tolérance aux fautes par exemple. Ainsi, toute classe peut être associée de façon déclarative à une classe de méta-niveau. Le langage doit en outre permettre d'exhiber les éléments internes de son fonctionnement qui seront gérés à un méta-niveau de programmation (l'état de l'objet, l'invocation de ses méthodes, des informations de type, etc.) — c'est la notion de "*réification*". Les caractéristiques du langage réifiées sont manipulées au méta-niveau. Un autre intérêt corollaire de cette approche serait la possibilité d'associer de façon dynamique objet et méta-objet (ce qui n'est pas le cas aujourd'hui) pour pouvoir adapter le comportement du système à de nouvelles configurations, et donc à de nouvelles hypothèses de fautes, à la migration d'objets dans de nouveaux environnements, en particulier mobiles.

Le développement d'applications distingue ainsi deux niveaux de programmation : le niveau de base (programmation fonctionnelle) où l'on déclare le modèle de fonctionnement interne de l'objet (son méta-objet associé), et le méta-niveau de programmation où l'on décrit le fonctionnement interne du modèle qui a été sélectionné. Les autres propriétés du modèle objet, telles que l'héritage, prennent ici tout leur sens, en particulier en ce qui concerne la programmation des mécanismes non fonctionnels, en facilitant la réutilisation de mécanismes classiques et leur adaptation à des besoins ou à des hypothèses particulières.

La programmation des objets (ou des classes qui leur correspondent dans un langage de classes) pose le problème de la représentation des objets à l'exécution. En effet, il s'agit ici d'objets actifs qui doivent

être représentés à un niveau d'abstraction inférieur en termes d'objets du système. La technologie micro-noyau semble être particulièrement adaptée à la représentation des objets (et des méta-objets) sous la forme d'unités d'exécution autonomes, et ce lorsque la granularité des objets considérés le permet. Pour ce qui concerne les objets de fine granularité, ils n'ont de notre point de vue pas d'existence réelle au niveau du support d'exécution. Le modèle client-serveur est une représentation possible au niveau système d'objets de forte granularité interagissant. Des modèles plus fins peuvent être définis au niveau d'environnements d'exécution orientés-objets bâtis sur ce type de plate-forme (COOL développé au dessus du micro-noyau Chorus, par exemple).

Plusieurs mécanismes de tolérance aux fautes par réplication dans un environnement réparti ont été expérimentés en utilisant cette approche [14] avec le langage Open-C++ (en fait un pré-processeur de C++). Des classes de méta-niveau permettant d'instancier un objet suivant plusieurs modèles de réplication ont été développées. Elles s'appuient sur l'interception de l'invocation de méthodes et sur l'accessibilité de l'état interne de l'objet au méta-niveau. Les classes de méta-niveau suivantes ont été mises en œuvre :

- *réplication passive* : le méta-objet correspondant permet la transmission transparente de points de reprise à une réplique passive;
- *réplication semi-active* : le méta-objet correspondant permet de synchroniser deux répliques actives dont une seule renvoie des réponses à l'objet client;
- *réplication active et vote majoritaire* : un méta-objet "serveur" permet l'exécution répliquée des méthodes ; un méta-objet "client" effectue d'invocation multiple des méthodes et effectue un vote majoritaire sur les résultats.

Dans les expérimentations actuelles, les interactions entre objets sont basées sur l'utilisation de "sockets". Dans le dernier exemple, la gestion de la distribution est effectuée au méta-niveau. L'invocation (répliquée) de méthodes est donc transparente au programmeur d'application.

3.1.4 Prospectives

Les prospectives actuelles consistent à étudier et à développer un environnement de programmation d'application fiables traitant de l'information confidentielle. Il s'appuie sur l'approche méta-objet que nous

avons décrite, ainsi que sur un environnement d'exécution sous-jacent basé sur la technologie micro-noyau. Les recherches actuelles s'effectuent selon trois axes principaux:

- Description formelle et analyse de la FRD

  La description formelle du processus de fragmentation des applications en suivant une approche objet sera poursuivie en prenant en compte les contraintes de confidentialité relatives à l'application. Des critères d'évaluation qualitatifs et quantitatifs devront être énoncés. Ces critères permettront de juger de l'intérêt de cette technique pour une application donnée, en particulier de comparer plusieurs organisations de l'application, en fonction de critères de performance.

- Environnement de programmation par méta-objets

  La mise en œuvre de la FRD par l'utilisation de la propriété de réflexivité est notre objectif. Le langage que nous avons choisi est Open-C++ qui offre la possibilité de redéfinir l'invocation de méthode et l'accès à un attribut ainsi que la création d'objet. Le développement de méta-objets sera poursuivi et enrichi : méta-objets basés sur les techniques de mémoire stable, utilisation de protocoles de diffusion atomique, utilisation de protocoles d'authentification et d'autorisation au méta-niveau, utilisation de techniques de chiffrement. Enfin, une hiérarchie d'héritage de classes au méta-niveau mêlant des aspects de tolérance aux fautes et de sécurité devra être définie et expérimentée.

- Environnement opérationnel

  Un système d'exécution réparti orienté-objet est sans nul doute primordial. L'idée consiste à définir un support d'exécution basé sur la notion de méta-objet dont le comportement soit adaptable à un environnement opérationnel (hypothèses de fautes, menaces). Le lien entre objet et méta-objet est alors nécessairement dynamique.

  Il semble qu'un tel environnement bâti sur micro-noyau soit la meilleure solution et c'est pour cette raison que nous avons décidé de mener nos études et nos expérimentations sur le micro-noyau Chorus. Des services de base devront être mis en œuvre pour la réalisation de méta-objets performants ; par exemple, les proto-

coles de diffusion atomique xAMp issus du projet DELTA-4 seront réutilisés et implémentés sur Chorus.

3.2 Protection et noyaux de sécurité

Participants : Yves Deswarte, Jean-Charles Fabre, Vincent Nicomette, David Powell

Nous avons initié cette année une nouvelle étude sur la protection. Classiquement, on définit la protection comme la fonction du système qui gère les droits d'accès de *sujets* (entités actives : processus, activités, etc.) sur des *objets* (contenant ou recevant de l'information), en fonction d'un politique de sécurité. Un sujet a un droit d'accès sur un objet s'il a le droit d'exécuter la fonction (ou méthode) correspondant à cet accès sur l'objet. La politique de sécurité est un ensemble de règles qui, si elles sont respectées, garantissent la sécurité-innocuité (*safety* en anglais) et/ou la sécurité-confidentialité (*security* en anglais).

La protection peut donc viser trois objectifs : empêcher des actions non-autorisées (malveillantes), limiter la propagation des erreurs, et détecter des erreurs, que ces erreurs soient dues à des fautes conception, à des fautes d'interaction ou à des fautes physiques.

En fait, au moins dans le cadre de notre étude, un sujet est une entité active, dynamique, qui s'exécute pour le compte d'un *utilisateur*, l'utilisateur étant une entité persistante représentant une personne physique ou un serveur (serveur d'impression, par exemple). Les droits sont donc affectés aux sujets en fonction d'informations persistantes, les *permissions*, correspondant aux privilèges des utilisateurs sur les objets persistants. Les permissions sont donc plutôt représentées comme des listes de contrôle d'accès, alors que les droits d'accès sont plutôt représentés comme des capacités.

Le projet SATURNE a déjà montré qu'il était possible de gérer les permissions au moyen d'un serveur de sécurité réparti tolérant à la fois les fautes accidentelles et les intrusions. Un tel serveur a été réalisé dans le cadre du projet ESPRIT DELTA-4. Nous nous intéressons maintenant davantage à la gestion des droits d'accès, basée sur la notion de *noyau de sécurité*. Un noyau de sécurité est un composant du système qui, s'il est *sûr*, garantit que la politique de sécurité est respectée. Ce noyau de sécurité sera développé au-dessus du micro-noyau Chorus, en mettant en œuvre des mécanismes généraux (labels, capacités) sur des objets de

fine granularité, tout en permettant une grande souplesse sur le choix de la politique de sécurité.

Nous avons élaboré un premier schéma d'autorisation mettant en œuvre des noyaux de sécurité dans les systèmes d'objets distribués [10]. Un système d'objets distribués est composé d'un ensemble d'objets dont la localisation est transparente aux utilisateurs. Certains de ces objets sont directement connus des utilisateurs (ce sont des objets persistants) et peuvent être invoqués grâce à des méthodes publiques. En revanche, il existe de nombreux objets temporaires, dédiés à la réalisation d'une action ponctuelle. De même, il existe des objets persistants mais dont la granularité est si fine que leur existence est totalement inconnue des utilisateurs d'un tel système.

La gestion des droits d'accès relatifs aux objets persistants et de forte granularité est prise en charge par un *serveur d'autorisation*. Le serveur d'autorisation est responsable de fournir aux utilisateurs des privilèges leur permettant d'accéder à des objets persistants. Ces privilèges seront délivrés seulement si l'utilisateur est autorisé à effectuer l'accès correspondant (le serveur d'autorisation gère une matrice d'accès). Sur chaque site du système, un noyau de sécurité est chargé de contrôler l'accès aux objets locaux. L'ensemble des noyaux de sécurité garantissent le respect de la politique de sécurité. Chacun de ces noyaux gère de façon autonome les droits d'accès aux objets temporaires et aux objets de fine granularité locaux.

Nous avons étudié ce schéma d'autorisation dans le cadre d'application d'une politique de sécurité multi-niveau basée sur des contrôles obligatoires. Nous avons pour cela attribué des labels aux utilisateurs, aux objets et aux requêtes du système. En effet, le mode de communication dans un tel système est la communication par envoi de messages (requêtes) et ces requêtes véhiculent de l'information sensible ; il est donc nécessaire de les étiqueter par des labels au même titre que les objets et les utilisateurs du système. Nous avons attribué à chaque utilisateur un label représentant son habilitation. En ce qui concerne les objets, nous avons distingué deux familles d'objets que nous avons étiquetés différemment. Certains objets échangent des informations avec les requêtes qui y accèdent ; nous avons attribué à ces objets un label (représentant la classification de l'information qui constitue leur état). En revanche, il existe des objets pour lesquels il n'y a aucun flux d'information entre eux et les requêtes qui y accèdent. Ces objets utilisent des informations

de la requête pour invoquer d'autres objets mais ne gardent aucune mémoire des ces informations, leur état est réinitialisé à chaque invocation. Nous avons attribué une parenthèse de labels à ces objets (deux labels représentant la confiance que l'on a dans de tels objets). Enfin, nous avons attribué aux différentes requêtes une parenthèse de labels. La politique de sécurité garantie par les noyaux de sécurité est basée sur des règles qui, en fonction des labels des requêtes et des labels des objets accédés par ces requêtes, autorisent ou non les accès et modifie ou non les parenthèses de chaque requête.

3.3 Évaluation quantitative de la sécurité

Participants : Marc Dacier, Yves Deswarte, Mohamed Kaâniche, Rodolphe Ortalo

Les travaux commencés en 1992 sur la problématique de l'évaluation quantitative de la sécurité des systèmes informatiques ont été poursuivis. Plus précisément, l'étude en cours vise à permettre d'identifier les vulnérabilités d'un système informatique, à pondérer ces vulnérabilités en fonction de l'effort nécessaire à un attaquant pour exploiter ces vulnérabilités, et enfin à évaluer globalement l'effort nécessaire à un attaquant pour mettre en défaut la politique de sécurité du système.

Schématiquement, cette approche est comparable à celle de l'évaluation de la fiabilité d'un système complexe, qui consiste à identifier les modes de défaillance des composants, à pondérer ces modes de défaillance par les taux de défaillance des composants en fonction du temps, puis à évaluer, en fonction de la structure du système, la fiabilité au sens probabiliste, c'est-à-dire la probabilité de bon fonctionnement en fonction du temps. Dans cette analogie, l'effort moyen nécessaire à un attaquant pour mettre en défaut la politique de sécurité correspond au temps moyen avant la première défaillance (MTFF : Mean Time to the First Failure).

Il est important de noter que l'objectif de l'étude est de fournir une estimation de la capacité d'un système à résister à une attaque et non pas une estimation de la probabilité qu'il soit attaqué. Ceci nécessiterait, en effet, une modélisation de la population d'attaquants et de leur stratégie de choix d'une cible, ce qui nous semble — à l'heure actuelle — irréalisable. Au contraire, notre étude ne s'intéresse qu'au système informatique et à ses utilisateurs, toutes choses pour lesquelles nous dis-

posons de données tangibles qui permettront de valider, dans une phase ultérieure, nos hypothèses de travail.

Cette approche est très différente des méthodes classiques d'évaluation de la sécurité qui reposent sur la vérification de la présence ou de l'absence de certaines caractéristiques fonctionnelles (authentification, droits d'accès, politique de sécurité, etc.) et de l'utilisation ou non de techniques de développement particulières (spécifications formelles, preuves, tests, etc.). Ces méthodes ont fourni la base de "critères d'évaluation de la sécurité" aux États-Unis (TCSEC : Trusted Computer System Evaluation Criteria), au Canada (The Canadian Trusted Computer Product Evaluation Criteria) et en Europe (ITSEC : Information Technology Security Evaluation Criteria). Mais ces méthodes "classiques" supposent une démarche incompatible avec les caractéristiques que l'on retrouve dans bon nombre de systèmes actuels : modification fréquente de la politique de sécurité, évolution rapide des configurations et des applications, ouverture sur des systèmes ou des utilisateurs inconnus, etc. En fait, dans de tels systèmes, les utilisateurs ne sont pas prêts, pour améliorer la sécurité, à renoncer à la facilité d'utilisation et de partage d'information.

Du fait de la plus grande liberté d'action laissée aux utilisateurs, la vision statique de la sécurité qu'offre l'analyse qualitative n'est que de peu d'utilité pour appréhender le niveau de sécurité opérationnelle. En revanche, l'évaluation quantitative de la sécurité permet de surveiller l'évolution de la sécurité en fonction de l'évolution des configurations et des applications, et de mesurer les améliorations de sécurité qui pourraient être obtenues au prix de légères modifications de configuration ou d'utilisation. Elle offre également à l'administrateur des données pragmatiques pour sensibiliser les utilisateurs au problème de la sécurité et, si nécessaire, pour les convaincre de modifier leur configuration de travail.

Notre démarche repose sur une modélisation du système informatique sous forme d'un graphe des privilèges [6]. Le graphe des privilèges est un modèle formel, basé sur le modèle de matrice d'accès typée de Ravi Sandhu, dans lequel les nœuds représentent des ensembles de droits (ou privilèges), et les arcs des transferts de privilèges : il existe un arc étiqueté M entre l'ensemble de droits A et l'ensemble B s'il est possible, ayant les droits de A d'acquérir les droits de B, en utilisant la méthode M. Cette méthode peut être un transfert licite de privilège (l'utilisateur ayant les privilèges B fait confiance à celui ayant les privilèges A), ou un

transfert implicite (B est un sous-ensemble de A), ou encore une attaque élémentaire. Un poids peut être affecté aux différentes méthodes, selon la difficulté pour un attaquant pour exploiter la méthode et/ou selon le temps nécessaire pour réaliser l'attaque. Ce graphe peut être analysé pour identifier les possibilités de mettre en défaut la politique de sécurité du système. En effet, à partir de la politique de sécurité, il est possible d'identifier des attaquants potentiels (intrus externes, ou utilisateurs pouvant tenter d'étendre leurs privilèges ou d'abuser de leurs privilèges), et des cibles potentielles (ensembles de droits d'accès à des informations sensibles). La politique de sécurité peut être mise en défaut s'il existe (au moins) un chemin depuis l'ensemble des privilèges d'un attaquant potentiel jusqu'à une cible.

Le graphe des privilèges peut être généré automatiquement, par exemple à l'aide de l'outil ASA, développé d'abord à l'Université de Louvain, qui permet d'analyser automatiquement la configuration d'un système Unix (ou d'un ensemble de machines Unix interconnectées) pour en déterminer les vulnérabilités (en fonction d'une liste de méthodes d'attaque connues), De même, à partir d'une description formelle de la politique de sécurité, il est possible d'identifier les nœuds correspondants à des attaquants potentiels, et ceux correspondant à des cibles potentielles.

A partir de ce graphe pondéré, il est possible d'évaluer la difficulté pour chaque attaquant potentiel d'atteindre chaque cible potentielle. Pour ce faire, nous transformons le graphe en un réseau de Petri stochastique généralisé afin de modéliser la logique du processus d'attaque. Les poids deviennent des taux de succès par attaque, fonction également de la cible. Ils sont une mesure de l'effort et du temps que l'attaquant doit dépenser pour réussir une attaque : le taux sera d'autant plus petit que l'effort nécessaire à la réussite sera grand.

On peut alors dériver le graphe des marquages correspondant à l'aide du logiciel SURF-2 conçu et développé au LAAS, qui sert de support pour calculer la distribution de probabilité d'atteindre une cible en fonction de l'effort à mettre en oeuvre. L'ensemble des résultats pour tous les couples (attaquant, cible) permet d'estimer sous forme quantitative la sécurité du système. Un prototype d'outil logiciel enchaînant ces différentes phases a été développé [11].

Ces travaux ont été marqués cette année par la soutenance de la thèse de Marc Dacier [1].

4 Actions industrielles

4.1 Le $\mathcal{LIS}$ (Laboratoire d'Ingénierie de la Sûreté de Fonctionnement)

Le $\mathcal{LIS}$ est un laboratoire de recherche commun créé en juin 1992 par Matra-Marconi-Space, Technicatome et le LAAS, avec le soutien de la Région Midi-Pyrénées. Il regroupe, dans les locaux du LAAS, des ingénieurs des partenaires industriels et des chercheurs du groupe "Tolérance aux Fautes et Sûreté de Fonctionnement Informatique". Le $\mathcal{LIS}$ comprend une dizaine de membres permanents auxquels s'adjoignent des doctorants.

L'objectif général du $\mathcal{LIS}$ peut se résumer par : "*Spécifier, concevoir, réaliser et exploiter des systèmes où la faute est naturelle, prévue et tolérable*". Afin de remplir cet objectif, les thèmes de recherche du $\mathcal{LIS}$ couvrent un domaine très vaste : processus de développement, cycle de vie, validation, étude des agressions externes, prise en compte des facteurs humains, procédures d'utilisation et de maintenance, documentation...

Le $\mathcal{LIS}$ se consacre à des recherches amont, tant théoriques qu'expérimentales, dont les résultats seront valorisés par les partenaires industriels. Ces recherches seront focalisées selon trois axes principaux :

- *Systèmes sûrs de fonctionnement* : prise en compte simultanée de trois attributs de la sûreté de fonctionnement : disponibilité, sécurité-innocuité, sécurité-confidentialité ; c'est principalement dans cet axe que sont impliqués les chercheurs du projet SATURNE ;
- *Logiciels sûrs de fonctionnement* : conception et validation de logiciels pour applications critiques ;
- *Interfaces homme-système sûres de fonctionnement* : tolérance aux fautes dues aux interactions humaines.

Ces trois axes de recherches sont complétés par la rédaction et le maintien à jour d'un *guide pratique de la sûreté de fonctionnement* [4].

Les travaux du $\mathcal{LIS}$ s'appliquent à de nombreux domaines, dont l'Espace et l'Énergie qui sont les principaux secteurs d'activité des partenaires industriels du $\mathcal{LIS}$, mais aussi les Télécommunications, le Transport et la Robotique.

4.2 Autres actions industrielles

Une collaboration a été établie avec l'Union des Assurances de Paris (UAP) pour la poursuite de l'étude sur l'évaluation quantitative de la sécurité.

Par ailleurs, une étude va être lancée avec Thomson-CSF et Chorus Systèmes en vue de l'amélioration de la sécurité du micro-noyau Chorus.

Jean-Charles Fabre a été consultant pour la société Chorus Systèmes sur les aspects d'injection de fautes.

Yves Deswarte a effectué une expertise pour l'ANVAR sur un projet de serveur de fichiers sécurisé proposé par la société Log'on.

5 Actions nationales et internationales

5.1 Actions nationales

Yves Deswarte est président du Comité Technique "Sécurité et sûreté informatiques" de l'AFCET. Ce comité technique comporte deux groupes de travail : "Sûreté de fonctionnement", dont Yves Deswarte, Jean-Charles Fabre et David Powell sont membres, et "Sécurité des systèmes d'information" dont Yves Deswarte est membre. Jean-Charles Fabre est membre de la commission "Enseignement" du Comité Technique "Sécurité et sûreté informatiques"; les travaux de cette commission portent sur des programmes de cours conjoints en sûreté de fonctionnement et sécurité informatique.

Par ailleurs, Jean-Charles Fabre est membre du groupe de travail de l'AFCET sur la "Technologie Objet". L'utilisation de la technologie objet pour la tolérance aux fautes en utilisant l'approche "Protocole à méta-objets" y a été présentée.

Le projet SATURNE participe au Pôle "Systèmes" du GDR "Parallélisme, Réseaux, Systèmes".

5.2 Actions internationales

Le groupe de recherche "Tolérance aux fautes et sûreté de fonctionnement informatique" du LAAS-CNRS est, avec l'Université de Newcastle (GB), l'un des principaux partenaires de l'Action de Recherche de Base

ESPRIT PDCS-2 (Predictably Dependable Computing System) qui regroupe également IEI-CNR (I), LRI (Orsay), City University of London (GB), l'Université Chalmers de Göteborg (S), l'Université de Vienne (A) et l'Université d'York (GB). Les membres du projet SATURNE y sont fortement impliqués, en particulier pour les aspects concernant les concepts de base de la sûreté de fonctionnement, l'évaluation et la tolérance aux fautes.

Toujours dans le cadre des Actions de Recherche de Base du programme ESPRIT, le groupe participe également au Réseau d'Excellence CaberNet (*Computer Architectures for Basic European Research Network*), qui regroupe, outre le LAAS, une quarantaine de centres de recherche académiques et industriels européens, dont l'INRIA (Rocquencourt et Rennes), les universités de Newcastle, Pise, Cambridge, Kaiserslautern, Twente et Vienne, le Trinity College de Dublin, l'INESC, Chorus, Bull-IMAG, etc.

Le projet SATURNE participe à une convention CNRS-NSF avec le "Laboratory for Dependable Computing and Fault Tolerance" de UCLA (Professor A. Avizienis).

6 Diffusion des résultats

6.1 Enseignement

Centre d'Études Supérieures de la Sécurité des Systèmes d'Information (CESSSI), Délégation Interministérielle à la Sécurité des Systèmes d'Information (DISSI), Issy-les-Moulineaux, Formation à la Sécurité des Systèmes d'Information (Yves Deswarte).

Université Paul Sabatier de Toulouse, Service de Formation Continue, Cours Système Informatique Unix (Jean-Charles Fabre).

Université Paul Sabatier de Toulouse, module ingénierie des protocoles de 3ème annee de l'IUP STRI (*Systèmes, Télécommunications et Réseaux Informatiques*), année 94–95, Cours Sécurité Informatique (Jean-Charles Fabre).

6.2 Participation à des conférences et des colloques

4th International Working Conference on Dependable Computing for Critical Applications, San Diego (USA), 4–6 Janvier 1994 : participa-

tion à deux tables rondes : "Qualitative vs. quantitative assessment of security" [7] (Marc Dacier) et "Common techniques in fault-tolerance and security" [8] (Yves Deswarte).

Workshop IFIP TC 10.4 et 11.3, San Diego (USA), 7–9 janvier 1994 : communication sur les failles de sécurité utiles (Yves Deswarte).

PDCS-2 Workshop, Brighton (GB), 2–4 février 1994 : présentation d'une approche de conception objet de traitements fragmentés et de mise en œuvre par méta-objets (Jean-Charles Fabre).

CaberNet Workshop, Dublin (Irlande), 23–25 mars 1994: présentation relative à l'utilisation d'une approche objet pour la conception d'applications tolérantes aux fautes et aux intrusions (Jean-Charles Fabre).

PDCS-2 Open Workshop, Newcastle (GB), 19–21 septembre 1994 : présentation d'une approche de mise en œuvre de mécanismes de tolérance aux fautes utilisant des langages orientés-objets réflexifs — Expérimentations de méta-objets en Open-C++ [14] (Jean-Charles Fabre).

1st European Dependable Computing Conference (EDCC-1), Berlin (Allemagne), 4–6 Octobre 1994 : communication sur une approche de conception orientée-objet de la fragmentation-redondance-dissémination appliquée au traitement d'informations confidentielles [9] (Jean-Charles Fabre).

OOPSLA'94 Conference. Workshop : Standards for Security in Object-Oriented Systems , Portland (USA), 23 Octobre 1994 : communication sur un schéma d'autorisation pour les systèmes à objets répartis [10] (Vincent Nicomette).

1st Smart Card Research and Advanced Application Conference (CARDIS'94), Lille (France), 24–26 Octobre 1994 : communication sur un serveur d'authentification réparti [5] (Laurent Blain) et participation à une table ronde sur la sécurité des systèmes nomades (Yves Deswarte).

European Symposium on Research in Computer Security (ESORICS 94), Brighton (UK), 7–9 novembre 1994 : communication sur la définition formelle du graphe des privilèges [6] (Marc Dacier).

6.3 Organisation de colloques et de cours

Yves Deswarte est le président du Comité de Pilotage International de l'European Symposium on Research in Computer Security (ESORICS). Il a été également membre du Comité de Programme d'ESORICS 94 qui s'est déroulé du 7 au 9 novembre 1994 à Brighton.

David Powell était le président et Yves Deswarte était membre du Comité de Programme de la 1ère European Dependable Computing Conference (EDCC-1) qui s'est tenue à Berlin du 4 au 6 octobre 1994.

Jean-Charles Fabre a été membre du comité de programme de la journée d'étude AFCET sur la protection des logiciels et des documents électroniques qui s'est tenue à Paris le 15 décembre 1994.

Yves Deswarte est membre du comité scientifique du Congrès AFCET 95 qui se tiendra à Toulouse du 25 au 27 octobre 1995, et Jean-Charles Fabre est membre du comité de programme du colloque "Technologie Objet" de ce congrès.

6.4 Diffusion de produit

Le LAAS-CNRS diffuse maintenant le logiciel SURF 2.

6.5 Autre

Conférence invitée sur la tolérance aux fautes et la sécurité, Télécom-Bretagne, Rennes, 10 mars 1994 (Yves Deswarte).

Séminaire du Laboratoire d'Ingénierie de la Sûreté de fonctionnement ($\mathcal{LIS}$), Carcassonne, 10–11 octobre 1994 : présentation de politiques de sécurité-confidentialité et de sécurité-innocuité (Yves Deswarte).

7 Publications

Thèses

[1] M. Dacier, *Vers une évaluation quantitative de la sécurite informatique*, Thèse de doctorat, Institut National Polytechnique de Toulouse, 20 décembre 1994.

Articles et chapitres de livre

[2] J. Arlat, Y. Deswarte, D. Powell, *ARAGO 15, Informatique Tolérante aux Fautes*, Masson, ISBN 2-225-84522-0, 1994, ch. Systèmes commerciaux, p. 73–79.

[3] J.-C. Laprie, J. Arlat, C. Beounes, K. Kanoun, *Encyclopedia of Software Engineering, Vol.1*, JJ.Marciniak, Ed.in Chief. Wiley Interscience, ISBN 0-471-54001-3, 1994, ch. Fault tolerant computing, p. 482–503.

[4] LIS, *Guide de la sûreté de fonctionnement*, 1994, 235 p., Rapport LAAS n° 94090.

Communications à des congrès, colloques, etc.

[5] L. Blain, Y. Deswarte, « A smartcard fault-tolerant authentication server », *in : 1st Smart Card Research and Advanced Application Conference (CARDIS'94)*, p. 149–165, Lille (France), 24-26 octobre 1994.

[6] M. Dacier, Y. Deswarte, « Privilege graph: an extension to the Typed Access Matrix model », *in : European Symposium on Research in Computer Security (ESORICS 94)*, Brighton (GB), 7-9 novembre 1994. Lectures Notes in Computer Science 875: Computer Security, ISBN 3-540-58618-0, 1994, Springer-Verlag, p. 319-334.

[7] M. Dacier, « A fault forecasting approach for operational security monitoring », *in : 4th International Working Conference on Dependable Computing for Critical Applications (DCCA-4)*, p. 142–143, San Diego (USA), 4-6 janvier 1994.

[8] Y. Deswarte, « Improving security by fault tolerance », *in : 4th International Working Conference on Dependable Computing for Critical Applications (DCCA-4)*, p. 254–255, San Diego (USA), 4-6 janvier 1994.

[9] J.-C. Fabre, Y. Deswarte, B. Randell, « Designing secure and reliable applications using FRS: an object-oriented approach », *in : 1st European Dependable Computing Conference (EDCC-1)*, Berlin (Allemagne), 4-6 octobre 1994. Lectures Notes in Computer Science 852: Dependable Computing, ISBN 3-540-58426-9, 1994, Springer-Verlag, p. 21-38.

[10] V. Nicomette, Y. Deswarte, « An authorization scheme for distributed object systems », *in : OOPSLA'94 Conference, Workshop 8: Standards for Security in Object-Oriented Systems*, p. 1–9, Portland (USA), 24-26 octobre 1994.

Rapports de recherche et publications internes

[11] C. ARNAUD, «Démonstrateur pour l'évaluation quantitative de la sécurite d'un système UNIX», *Rapport de recherche n°94322*, LAAS-CNRS, septembre 1994, 90p.

[12] M. DACIER, Y. DESWARTE, «Propagation of privileges and security trade-off», *Rapport de recherche n°94031*, LAAS-CNRS, fevrier 1994, 14p.

[13] J.-C. FABRE, Y. DESWARTE, L. BLAIN, «Fault-tolerance and security by fragmentation-redundancy-scattering», *Rapport de recherche n°93472*, LAAS-CNRS, octobre 1994, 20p.

[14] J.-C. FABRE, V. NICOMETTE, T. PERENNOU, Z. WU, «Implementing fault-tolerant applications using reflective object-oriented programming», *2nd year report*, Contrat ESPRIT III Esprit Basic Research Action n° 6362 : PDCS-2, juillet 1994, Rapport LAAS n° 94156, 22p.

[15] F. PEIGNE, «Quelques environnements micro-noyau pour l'intégration de servies tolérant les fautes et les intrusions», *Rapport de stage*, DEA, Institut National Polytechnique, Toulouse, septembre 1994, 46p.

[16] E. TOTEL, F. PEIGNE, «Mécanismes répartis de tolérance aux fautes et aux intrusions : intégration sous Unix», *Rapport de stage de 3ème année*, ENSEEIHT, Toulouse, juin 1994, 79p.

[17] E. TOTEL, «Quelques protocoles de diffusion atomique pour l'implémentation de services répartis tolérant aux fautes», *Rapport de stage*, DEA, Institut National Polytechnique, Toulouse, septembre 1994, 47p.

8 Abstract

The SATURNE project aims to provide a global approach for the tolerance of accidental faults and intentional interaction faults (i.e., intrusions) in distributed systems. This approach enables to tackle reliability and security problems to be added in a unified way.

Within the scope of SATURNE project, a novel method has been developed to tolerate faults while preserving confidentiality. This method is called "fragmentation-redundancy-scattering"[9] [13].

The fragmentation-redundancy-scattering (FRS) technique consists in splitting information into fragments such that individual fragments do not carry significant information, adding redundancy to the fragments in order to tolerate deletion or modification of some fragments, then scattering these fragments throughout the distributed system. This

technique is viewed as complementary to existing security techniques, such as protection and cryptography.

FRS has first been applied, within the ESPRIT DELTA-4 projet, to a distributed file storage server and to a distributed security server. Today, this work is being extended to improve the reliability of processing of confidential information (see Section 3.1).

In parallel, a study is dedicated to fine-grain object protection and security and/or safety kernels (see Section 3.2). Another study is continuing on quantitative evaluation of security (see Section 3.3).

Table des matières