Projet SATURNE

Système réparti tolérant les fautes et les intrusions

Localisation : *Toulouse*

Mots-clés : sûreté de fonctionnement, système réparti, tolérance aux fautes, sécurité informatique, technologie à objets.

1 Composition de l'équipe

Responsable scientifique

Yves Deswarte, DR INRIA

Personnel INRIA

Jean-Charles Fabre, CR

Personnel LAAS-CNRS[1]

Mohamed Kaâniche, CR
David Powell, DR

Chercheurs doctorants

Marc-Olivier Killijian, boursier CNRS
Vincent Nicomette, boursier $\mathcal{LIS}$
Rodolphe Ortalo, boursier CNRS-UAP
Tanguy Pérennou, boursier MESR
Frédéric Salles, boursier MESR
Eric Totel, boursier CNRS

Autres personnels

Françoise Cabrolié, stagiaire CNAM
Cyril Delamare, stagiaire ENSEEIHT
Jacques Laffont, stagiaire DEA Université Paul Sabatier
Frédéric Melle, stagiaire IIE

[1] Temps partiel

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 contrepartie 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 [356], qui a été depuis reprise par un certain nombre de laboratoires académiques et industriels. 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. Après avoir été appliqués à un service réparti d'archivage de fichiers et à la gestion de la sécurité dans un système réparti, ces travaux ont été développés pour améliorer la fiabilité des traitements portant sur des données confidentielles.

Ceci nous a conduits à nous intéresser à l'implémentation de caractéristiques non-fonctionnelles dans les systèmes orientés-objets, et en particulier aux protocoles à méta-objets (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 et développés (cf. §3.3).

3 Action de recherche

3.1 Architectures à métaobjets et tolérance aux fautes

Participants : Jean-Charles Fabre, Tanguy Pérennou, Françoise Cabrolié, Cyril Delamare, Marc-Olivier Killijian, Arnaud Ladrech, Frédéric Melle, Frédéric Salles

Le développement d'applications tolérantes aux fautes nécessite la définition et l'intégration aisée de mécanismes permettant de mettre en œuvre des caractéristiques méta-fonctionnelles (redondance, authentification, chiffrement, communication à distance). L'utilisation de techniques de développement orienté-objet présente de multiples intérêts, tant au niveau de la conception que de la mise en œuvre. En particulier, la propriété de réflexivité de certains langages à objets présente un triple avantage :

- séparation entre développement de l'application elle-même (niveau de base) et développement des mécanismes de tolérance aux fautes à des niveaux d'abstraction différents (métaniveaux) ;
- définition, à chaque niveau d'abstraction, de mécanismes de tolérance aux fautes en utilisant les propriétés classiques des langages à objets, telles que la spécialisation par héritage, le polymorphisme, la liaison dynamique, etc.
- intégration transparente dans les applications des caractéristiques méta-fonctionnelles.

Le développement d'applications réparties mêlant des mécanismes de tolérance aux fautes et de sécurité-confidentialité en suivant cette approche fait l'objet de nos travaux actuels. Nous l'avons abordé sous deux angles :

- utilisation la propriété de réflexivité pour intégrer de façon transparente des mécanismes de tolérance aux fautes dans des applications ;
- définition d'une hiérarchie d'héritage permettant de concevoir de mécanismes de tolérance aux fautes en réutilisant (spécialisant) des classes de base.

Un modèle d'architecture de système basé sur une utilisation récursive des métaobjets a été défini. L'expérimentation de cette architecture a été effectuée sur un réseau de stations Unix et aussi sur un réseau de stations Chorus. Le protocole à métaobjet utilisé est celui d'Open C++ et les protocoles de groupe utilisés sont ceux fournis par le logiciel xAMp. Nous avons pu constater la souplesse d'intégration procurée par cette approche, en premier lieu, l'indépendance entre la programmation de l'application et la programmation des mécanismes utilisés, et aussi la possibilité de changer de mécanisme sans incidence sur le code source de l'application. Ces prototypes proposent sous forme de métaobjets différents types de mécanismes de tolérance aux fautes (mécanismes à support stable et différents mécanismes de réplication) ainsi que des mécanismes de sécurité (authentification, chiffrement). Une application répartie de type bancaire a été développée pour évaluer la flexibilité de l'approche. Tous les cas de composition possibles des métaobjets existant à différents métaniveaux ont été testés avec succès.

Les performances du système ont pu être mesurées et ont montré que l'interaction objet-métaobjet était négligeable par rapport aux fonctionnalités de tolérance aux fautes proprement dite (coût des communications de groupe par exemple).

La conception des mécanismes de tolérance aux fautes a été effectuée en utilisant la méthode de développement orienté-objets BON et a permis de définir une hiérarchie de classes (au sens de l'héritage) en vue de leur réutilisation. Une première hiérarchie de mécanismes de tolérance aux fautes a été définie : elle a montré que l'on pouvait factoriser ces mécanismes et donc favoriser la réutilisation de classes de base et leur spécialisation.

Une analyse des dépendances entre métaniveaux (partage d'information, par exemple), des limites de l'approche à métaobjet ainsi que de son utilisation à plusieurs niveaux a été effectué. Elle a montré que si l'indépendance des métaniveaux pouvait être obtenue à l'utilisation, il n'y avait pas indépendance totale lors de la conception des métaobjets. En particulier il existe une dépendance forte entre les mécanismes répartis de tolérance aux fautes et la gestion des communication de groupe. La paramétrisation du metaniveau de communication a été étudiée et limités à quelque variables partagées.

Ces travaux et les résultats obtenus ont été concrétisés dans la thèse de Tanguy Pérennou.

Les prospectives de ce travail concernent l'évaluation de cette approche ainsi que son extension à des support d'exécution à objet standard d'une part et à d'autres caractérisques méta-fonctionnelles. Un souci tout particulier sera porté sur la réutilisation des mécanismes, l'étude d'un protocole à métaobjet plus efficace et moins figé.

Le fait de disposer aujourd'hui d'un prototype complet du système proposant au programmeur un ensemble de bibliothèques de classes de métaobjets permet d'évaluer l'approche selon différents points de vue :

- facilité d'utilisation dans des applications de plus grande envergure ;
- réutilisation des mécanismes existants et développement de variantes ;
- protocole à métaobjet permettant de lier dynamiquement objet et métaobjet ;
- définition d'un protocole à métaobjet pour différentes caractéristiques méta-fonctionnelles.

Enfin, de premières investigations sur l'utilisation de support d'exécution répartis d'objets, tel que COOL-ORB (au standard CORBA), l'intégration d'un protocole à métaobjet dans ce type de support seront engagées. L'utilisation d'autres langages de programmation et aussi la prise en compte de modèles de calcul moins contraignants sera étudiée (primitives d'intéraction asynchrone, concurrence intra-objet). Enfin, nous étudions aussi des mécanismes d'injection de faute par logiciel dans des micro-noyaux pour analyser leur comportement en présence de fautes; la définition des mécanismes de tolérance aux fautes en dépendent fortement. Des expérimentations seront menées sur Chorus.

3.2 Protection et noyaux de sécurité

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

Après avoir réalisé un serveur de sécurité réparti tolérant les fautes accidentelles et les intrusions 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 met 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 [361, 360, 355]. 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.

Le schéma d'autorisation met en œuvre deux niveaux de protection. 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 la fourniture 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). Nous avons donné à ce serveur d'autorisation une grande souplesse quant à la gestion des droits d'accès, tout en garantissant au mieux le principe du moindre privilège.

Ce schéma d'autorisation est adaptable à différentes politiques de sécurité. En particulier, nous avons développé une politique obligatoire multi-niveau permettant de garantir des propriétés de confidentialité. Cette politique est une adaptation au modéle objet de la politique de Bell et LaPadula, et comme elle assigne des labels aux différentes entités du systèmes : habilitation pour les utilisateurs, classification pour les objets à état, intervalle de confiance pour les objets sans état et parenthèse pour les activités. La politique ainsi obtenue est plus souple que celle de Bell-LaPadula, en ce qu'elle autorise des flux d'information légitimes qui seraient interdits par une application directe de la politique de Bell-LaPadula. Mais elle est tout aussi efficace, puisqu'on peut montrer qu'elle interdit bien tous les flux d'information illégitime.

Un prototype de serveur d'autorisation et de noyau de sécurité a été réalisé sur un système réparti composé de micro-ordinateurs avec le système à micro-noyaux Chorus. Une démonstration, mettant en œuvre un système de messagerie multi-niveau a été développée et est aujourd'hui opérationnelle.

Un autre schéma d'autorisation est en cours de développement qui concerne cette fois la préservation de l'intégrité d'un système orienté objet, et en particulier adresse le problème de la coexistence de logiciels de criticité différente. En effet, dans la plupart des applications coexistent des fonctions

critiques (par exemple le pilotage d'un avion) et des fonctions non-critiques (par exemple la gestion de la vidéo des passagers). Dans la mesure où les fonctions non-critiques peuvent perturber (en monopolisant les ressources) ou contaminer (par propagation d'erreurs) les fonctions critiques, la solution industrielle actuelle consiste à appliquer aux logiciels non-critiques les méthodes de développement et de vérification exigées pour les logiciels critiques (normes ou standards propres à chaque domaine d'application tel que l'avionique, le nucléaire, le ferroviaire...). On peut envisager au contraire d'autoriser la coexistence de logiciels validés à des niveaux différents, en utilisant des mécanismes de protection adéquats.

C'est ce que propose notre politique multi-niveau inspirée de la politique de Biba et adaptée à un environnement objet. Chaque objet du système se voit associer un niveau d'intégrité (en relation étroite avec le degré de criticité de ce dernier, et par conséquent avec son niveau de validation) qui caractérise la confiance qu'on porte en son comportement. Notre schéma d'autorisation permet le contrôle des flux d'informations entre les différents objets du système, le but étant d'éviter toute propagation d'erreurs d'un objet de faible intégrité vers un objet de plus haute intégrité.

La solution proposée permet de garantir une souplesse plus grande d'utilisation que celle de Biba grâce à l'introduction de différents types d'objets additionnels dans le système. En effet, on compte d'une part des objets mono-niveaux qui obéissent aux règles strictes de Biba, mais également des objets multi-niveaux et des objets dits de validation. Les objets multi-niveaux ont la propriété de pouvoir parcourir plusieurs niveaux sans pour autant autoriser des flux illégaux, ce qui en rend l'utilisation plus souple (bien que des règles de validation propres doivent leur être appliquées). Les objets de validation utilisent des mécanismes de tolérance aux fautes permettant la remontée d'informations de faible niveau d'intégrité vers des niveaux de plus haute intégrité (leur rôle étant en fait de faire monter la confiance que l'on porte aux données au niveau nécessaire); ce flux normalement interdit est un apport essentiel à ce type de politique, proposant une solution au problème classique de la dégradation du niveau des informations.

3.3 Evaluation quantitative de la sécurité

Participants : Yves Deswarte, Mohamed Kaâniche, Rodolphe Ortalo

La poursuite des travaux relatifs à l'évaluation quantitative de la sécurité des systèmes s'est concentrée sur deux axes. D'une part, la mise en œuvre expérimentale de la méthode d'évaluation sur un système informatique de taille importante a été effectuée sur une longue période grâce au prototype existant. Ceci a permis de rassembler une grande quantité de résultats et d'évaluer la capacité de la méthode à fournir une évaluation pertinente de la sécurité. D'autre part, un formalisme de spécification des objectifs de sécurité d'un système a été défini et expérimenté par le biais de l'implémentation d'un prototype d'éditeur de politiques de sécurité. Un modèle des vulnérabilités du système (par exemple un graphe des privilèges), joint à la description de ses objectifs, fournit les éléments permettant la mise en œuvre de la méthode d'évaluation, et laisse penser qu'elle peut être étendue des systèmes informatiques aux systèmes d'information et aux organisations.

Afin de spécifier les objectifs de sécurité, ainsi que les mécanismes contenus dans le système qui sont liés à la sécurité, nous avons étudié la possibilité d'utiliser un formalisme logique, la logique modale, et plus précisément une de ses branches, la logique déontique. L'atout principal de ce formalisme est de faire coexister dans un même langage logique les propositions usuelles de la logique propositionnelle et la notion plus riche de nécessité (et son dual, la possibilité) susceptible de prendre une sémantique d'obligation (et de permission) sans préjuger de la description d'autres propriétés du système. Ce formalisme semble particulièrement adapté pour l'écriture formelle d'une politique de sécurité, son expressivité permet en effet de combiner des opérateurs exprimant des contraintes fortes sur le système décrit (plus précisément ses objectifs de sécurité) et la description plus traditionnelle du fonctionnement

des mécanismes de sécurité et les propriétés de l'état de protection réel (éventuellement imparfait du point de vue des objectifs de sécurité). L'expressivité de ce langage laisse de plus supposer qu'il puisse être utilisé pour décrire des systèmes d'informations plus généraux que les systèmes informatiques et s'appliquer à des organisations. Afin d'expérimenter ce formalisme, un prototype d'éditeur de politique de sécurité, utilisant une représentation graphique des symboles du langage a été développé et utilisé pour représenter des mécanismes de sécurité, tout d'abord relatifs à des systèmes informatiques [369]. Cette expérimentation a permis de valider la faisabilité d'une telle approche en montrant qu'il était relativement aisé de représenter et de structurer à la fois les objets et les mécanismes relatifs à la sécurité d'un système, et les propriétés que l'on désire voir satisfaites. L'analyse des objectifs de sécurité ainsi spécifiés permet d'identifier les sujets et les objets qui sont plus particulièrement relatifs à la vérification d'un objectif de sécurité précis. Par exemple dans le cas d'un système informatique, on peut ainsi déterminer les utilisateurs dont les données doivent être protégées, ainsi que ceux qui sont susceptibles de les menacer. L'utilisation d'un modèle représentant les vulnérabilités du système permet alors de savoir quelles sont les classes de mécanismes utilisables pour mettre en défaut un objectif de sécurité particulier.

Notre démarche repose sur une modélisation des vulnérabilités du système informatique sous forme d'un graphe des privilèges. Le graphe des privilèges est un modèle formel dans lequel les nœuds représentent des ensembles de droits (ou privilèges), et les arcs des transferts de privilèges. Ces transferts peuvent être licites (un utilisateur ayant les privilèges B fait confiance à celui ayant les privilèges A), ou implicites (B est un sous-ensemble de A), ou encore illicites (le transfert est une attaque élémentaire). Un poids peut être affecté à chaque transfert, selon la difficulté et/ou le temps nécessaire pour un attaquant pour exploiter la méthode correspondant au transfert élémentaire de privilèges. Ce graphe peut être analysé pour identifier les possibilités de mettre en défaut les objectifs de sécurité du système. En effet, à partir de la politique de sécurité, il est possible d'identifier les attaquants potentiels, internes ou externes, et les cibles potentielles (ensembles de privilèges 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 jusqu'à une cible. A partir de ce graphe pondéré, il est possible d'évaluer la sécurité comme étant la difficulté pour les attaquants potentiels d'atteindre les cibles potentielles. L'évaluation quantitative s'appuie alors sur l'exploitation numérique du MTFF calculable à partir de ce réseau.

Dans le cas d'un système informatique, le graphe des privilèges peut être généré automatiquement, par exemple à l'aide d'un outil permettant d'analyser automatiquement la configuration d'un système Unix pour en déterminer les vulnérabilités. De même, une description formelle des objectifs de sécurité permet d'identifier les cibles et les attaquants potentiels. Le prototype d'outil logiciel ESOPE enchaînant les différentes phases de l'évaluation a été utilisé pour collecter des données pendant plus d'un an sur un système informatique d'une centaine de machines et de plusieurs centaines d'utilisateurs. L'évaluation quantitative effectuée sur ces données pour un certain nombre d'objectifs de sécurité simple (protection du super-utilisateur ou d'un groupe particulier d'utilisateurs au sens d'Unix) et les résultats obtenus ont montré que l'évolution de la mesure permet d'identifier un certain nombre d'événements pertinents pour la sécurité et liés à la vie opérationnelle du système [368].

4 Actions industrielles

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

Le $\mathcal{LIS}$ a été fondé en Juillet 1992 sous la forme d'un laboratoire commun entre le LAAS, Matra Marconi Space et Technicatome, avec le soutien de la Région Midi-Pyrénées. Implanté au LAAS, le $\mathcal{LIS}$ est constitué de chercheurs du groupe Tolérance aux Fautes et Sûreté de Fonctionnement Informatique, et d'ingénieurs détachés par les partenaires industriels.

L'objectif global du $\mathcal{LIS}$ peut se résumer par :"*Spécifier, concevoir, réaliser et exploiter des systèmes où la faute est naturelle, prévue et tolérable*".

Le $\mathcal{LIS}$ se consacre à des recherches amont, tant théoriques qu'expérimentales, dont les résultats sont valorisés par les partenaires industriels. Dans sa première phase (1992-1996), le $\mathcal{LIS}$ menait ses travaux selon trois axes principaux :

- *Systèmes sûrs de fonctionnement* : prise en compte simultanée de trois attributs de la sûreté de fonctionnement : disponibilité, sécurité - innocuité et sécurité - confidentialité,
- *Logiciels sûrs de fonctionnement* : conception et validation de logiciels pour applications critiques,
- *Interfaces homme-système sûres de fonctionnemen* : tolérance aux fautes dues aux interactions humaines.

Les trois axes de recherche du $\mathcal{LIS}$ sont complétés par la rédaction et le maintien à jour d'un guide de la sûreté de fonctionnement. L'ouvrage résultant a été publié en 1995 chez Cépaduès-Editions et une deuxième édition (augmentée) a paru en 1996 [354].

Le succès des travaux menés par le $\mathcal{LIS}$ a conduit les premiers partenaires à souhaiter vivement la poursuite du laboratoire au-delà de sa durée initialement prévue de quatre ans. Par ailleurs, le $\mathcal{LIS}$ a été l'objet depuis sa création de demandes de la part d'autres partenaires potentiels désirant se joindre au laboratoire. Ceci a conduit à élargir la composition du $\mathcal{LIS}$ en accueillant trois nouveaux partenaires : Thomson-CSF, l'Aérospatiale et EDF, sous forme d'un "laboratoire coopératif".

La considération conjointe des résultats obtenus depuis la création du $\mathcal{LIS}$ et des nouveaux défis à prendre en compte conduit à structurer les activités du $\mathcal{LIS}$ renouvelé selon les trois axes suivants :

- *Réutilisation pour la sûreté de fonctionnement* : vérification et évaluation pour des contextes d'utilisation différents ; approches orientées-objet.
- *Protection dans les systèmes sûrs de fonctionnement* : schémas d'autorisation ; coexistence de logiciels de criticités différentes ; tolérance aux erreurs d'interaction.
- Conception de systèmes sûrs de fonctionnement en vue de leur validation : répartition des tâches entre le système et ses opérateurs ; modélisation fonctionnelle et comportementale en présence de fautes ; modèle de développement de systèmes à sûreté de fonctionnement explicite.

La participation de Saturne au $\mathcal{LIS}$ concerne principalement la protection dans les systèmes sûrs de fonctionnement.

5 Actions nationales et internationales

5.1 Actions nationales

Les membres du projet Saturne sont fortement impliqués dans le 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".

Le projet SATURNE participe au Pôle "Systèmes d'exploitation" du GDR-PRC "Parallélisme, Réseaux, Systèmes".

5.2 Actions internationales

Le projet de recherche à long terme ESPRIT Design for Validation (DeVa) regroupe : City University of London (GB), Defense Research Agency (GB), LAAS-CNRS, l'École Polytechnique Fédérale de Lausnne (CH), l'Université de Newcastle (GB), l'Université d'Ulm (DE) et l'Université d'York (GB), plus trois membres associés : AIB Vincotte Nuclear (Belgique), LRI-CNRS et Université de Paris-Sud et l'Université Technique de Vienne (AT). La direction du projet est assurée conjointement par l'Université de Newcastle (principal contractant) et par le LAAS-CNRS. Le projet DeVa vise à aider à résoudre les problèmes de validation de systèmes informatiques critiques. DeVa s'attache particulièrement à la validation du logiciel vis-à-vis des exigences de sûreté de fonctionnement plutôt que vis-à-vis des exigences fonctionnelles, en insistant sur les aspects de structuration du logiciel pour concevoir en vue de leur validation des systèmes répartis temps-réel. Pour cela, la recherche se focalisera sur un ensemble de thèmes étroitement reliés concernant la spécification, la conception, l'implémentation, la vérification et l'évaluation. DeVa s'intéresse à des problèmes et des solutions potentielles applicables de façon générale plutôt que de se concentrer sur un secteur industriel particulier ou sur des types particuliers de systèmes informatiques (par exemple, temps réel dur ou souple), ou sur des attributs particuliers de la sûreté de fonctionnement. Dans ce projet, la participation du projet Saturne concerne principalement la conception de systèmes sûrs de fonctionnement à base de protocoles à méta-objets et l'évaluation quantitative de la sécurité opérationnelle. Le projet DeVa a démarré le 15 décembre 1995 pour une durée prévue de 3 ans.

Le projet de recherche précompétitive ESPRIT GUARDS (Generic Upgradable Architecture for Real-time Dependable Systems) regroupe trois industriels en pointe dans les domaines du nucléaire, du transport ferroviaire et de l'espace, Technicatome (principal contractant), Ansaldo Transporti (IT) et Matra Marconi Space (FR) ainsi qu'Intecs Sistemi (IT), le LAAS-CNRS, Pisa Dependable Computing Centre (IT), Siemens AG Osterreich PSE (AT) et l'Université d'York. Le projet GUARDS vise à développer des architectures, des méthodes, des techniques et des outils pour faciliter la conception, l'implémentation et la validation de systèmes temps-réel critiques. Pour atteindre cet objectif stratégique, une approche innovante est sous-tendue par l'élaboration et l'exploitation d'une méthodologie cohérente intégrant cinq caractéristiques fondamentales :

- la généricité, pour permettre la réutilisation de composants matériels et logiciels et d'architectures dans des applications et des domaines multiples, en particulier dans l'espace, le nucléaire et le transport ferroviaire ;
- la sûreté de fonctionnement, pour sous-tendre la conception et l'implémentation de mécanismes de tolérance aux fautes adaptés aux exigences de sécurité et de disponibilité ;
- le temps-réel, pour permettre de satisfaire des contraintes sévères de temps-réel imposées par les applications considérées ;
- l'aptitude à la validation, en permettant la définition d'instances particulières de l'architecture générique s'adaptant au mieux aux exigences imposées par chacones des applications visées ;
- l'aptitude à la certification, pour satisfaire les obligations de certification imposées par les diverses autorités de normalisation.

L'une des bases de cette architecture générique est l'intégration des politiques d'intégrité multi-niveaux prenant en compte les mécanismes de tolérance aux fautes pour permettre la coexistence de logiciels

de criticiés différentes. Ceci correspond à un des axes actuels du projet SATURNE.

Le projet GUARDS a démarré en février 1996 pour une période de trois ans.

Le projet de recherche SQUALE (Security, Safety and Quality Evaluation for Dependable Computing Systems) est un projet du programme européen ACTS. SQUALE regroupe trois sociétés spécialisées dans l'évaluation de la sécurité : CR2A-DI (contractant principal, FR), Admiral (GB), IABG (DE), deux industriels ayant des applications critiques : le CEA et Matra Transports, et un laboratoire de recherche, le LAAS. L'objectif du projet est d'étendre les méthodes d'évaluation basées sur des "critères d'évaluation", classiques dans le domaine de la sécurité-confidentialité, à d'autres attributs de la sûreté de fonctionnement, en particulier la sécurité-innocuité et la disponibilité. Parmi les critères d'évaluation de la sécurité-innocuité, les plus connus sont ceux du "livre orange" ou TCSEC (Trusted Computing System Evaluation Criteria), définis par la défense américaine il y a une dizaine d'années, et plus récemment ceux des ITSEC (Information Technology Security Evaluation Criteria) soutenus par les Communautés Européennes, mais d'autres ont été proposés par le Canada, le Japon et il y a quelques années par le gouvernement fédéral américain. Une harmonisation de ces différents critères est en cours et une ébauche de "Critères Communs" a été publiée et devrait conduire à une norme internationale. Tous ces critères visent à évaluer la sécurité-confidentialité ("security" en anglais), en vérifiant que les systèmes possèdent les caractéristiques leur permettant de satisfaire les exigences de sécurité (critères de fonctionnalité) et en vérifiant que ces caractéristiques ont été implémentées de façon adéquate (critères d'assurance). La même approche peut être utilisée pour évaluer d'autres caractéristiques de sûreté de fonctionnement, comme la sécurité-innocuité ("safety" en anglais) ou la disponibilité. L'objectif de ce projet est donc de définir des critères suffisamment génériques pour prendre en compte les exigences de sûreté de fonctionnement de différents secteurs d'application, et en particulier de permettre d'évaluer des systèmes ayant des exigences à la fois de sécurité-innocuité et de sécurité-confidentialité. Ces critères génériques doivent être compatibles avec les normes et les méthodes de certification qui ont cours dans les différents secteurs ayant des applications critiques (par exemple, le transport aérien ou ferroviaire, le nucléaire, les applications médicales, etc.), et doivent faciliter la certification par les autorités de ces différents secteurs. Les critères que nous proposons seront validés par une expérimentation portant sur des systèmes critiques proposés par le CEA et Matra Transports. Le projet SQUALE a démarré en mars 1996 pour une période de deux ans.

5.3 Actions diverses

Participation au jurys de thèse d'Yves BERGEON, Les canaux cachés dans les protocoles réseaux, Thèse de Doctorat de l'Université de Paris XI, 15 mai 1996 (Yves Deswarte, rapporteur et membre du jury).

6 Diffusion des résultats

6.1 Actions d'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 année de l'IUP STRI (*Systèmes, Télécommunications et Réseaux Informatiques*), année 95-96, *Cours Sécurité Informatique*

(Jean-Charles Fabre).

École des Mines de Nantes, séminaires *Sûreté de fonctionnement et tolérance aux fautes* et *Sécurité Informatique* de 3ème année, option *Qualité et Sûreté de Fonctionnement* (Yves Deswarte, Jean-Charles Fabre).

École Nationale Supérieure d'Ingénieurs de Construction Aéronautique (ENSICA) année de spécialisation en systèmes informatiques, *Sûreté de Fonctionnement et Tolérance aux Fautes* et *Sécurité Informatique* (Yves Deswarte, Jean-Charles Fabre).

École Nationale de l'Aviation Civile (ENAC), cours aux ingénieurs du Centre d'Études de la Navigation Aérienne, *Sûreté de Fonctionnement et Tolérance aux Fautes* (Yves Deswarte).

Caisse Nationale d'Assurance Maladie, cours *Sécurité et Cryptographie* (Yves Deswarte).

6.2 Participation à des colloques

IFIP WG 10.4 Workshop, La Martinique (FR), 23-25 janvier 1996, présentation *"Implementing Some Distributed Fault-Tolerant Mechanisms using Object-Oriented Programming"* (Jean-Charles Fabre).

Journée *Carrefour Recherche et Développement Technologique*, Conseil Régional, Toulouse, 2 février 1996, présentation *"Utilisation de la technologie des objets: Intérêts, exemples et interrogations"* (Jean-Charles Fabre).

Séminaire du Laboratoire d'Ingénierie de la Sûreté de fonctionnement ($\mathcal{LIS}$), 13-14 février 1996 à La Grande Motte, présentation d'une communication *"Protection des systèmes à objets répartis"*, (Yves Deswarte).

1st Closed Workshop of the DeVa Project, 13-15 mars 1996 à Crans-Montana, Suisse, présentation de trois communications *"Quantitative Evaluation of Operational Security"*, présentation *"Implementing Some Distributed Fault-Tolerant Mechanisms using Object-Oriented Programming"* (Jean-Charles Fabre) et *"SQUALE"* (Yves Deswarte).

Séminaire du Laboratoire d'Ingénierie de la Sûreté de fonctionnement ($\mathcal{LIS}$), 29 mars 1996 à Toulouse, présentation d'une communication *"Schémas d'autorisation et co-existence de logiciels de criticités différentes"* (Yves Deswarte).

Reflection'96, San Francisco (US) 21-23 avril 1996, Participation à la table ronde *"Designing Reflective Facilities"* (Jean-Charles Fabre).

AFCET, Groupe Technologie Objet, Journée *"Technologie Objet et Sûreté de Fonctionnement"*, 25 avril 1996, INSTN-CEA, Centre d'Etudes de Saclay, présentation de l'arcticle *"Mécanismes répartis de tolérance aux fautes : Implémentation par programmation orientée-objet"* (Jean-Charles Fabre).

DeVa Workshop, Schloss Reisensburg (DE), 18-20 septembre 96, présentation *"A system architecture and some experiments using metaobjects"* (Jean-Charles Fabre).

ESORICS'96, *4th European Symposium on Research in Computer Security*, présentation de [361] (Vincent Nicomette).

Yves Deswarte a été membre du Comité de Programme de la 2ème *European Dependable Computing Conference (EDCC-2)* qui s'est tenue du 2 au 4 octobre 1996 a Taormina (IT).

EDCC-2, Taormina (IT), 2-4 octobre 1996, présentation de l'article [359] (Jean-Charles Fabre).

Séminaire du Laboratoire d'Ingénierie de la Sûreté de fonctionnement ($\mathcal{LIS}$), 20-21 novembre 1996 à Cadarache, présentation d'une communication *"Tolérance aux malveillances"* (Yves Deswarte).

ASIAN'96, *2nd Asian Computing Science Conference*, présentation de [360] (Yves Deswarte).

6.3 Conférences invitées, tutoriels, cours, etc.

UCLA (US) 17-18 Avril 1996, *"Implementing Some Distributed Fault-Tolerant Mechanisms using Object-Oriented Programming"* (Jean-Charles Fabre).

Université de Valence (ES), 8 Novembre 1996, présentation *"A Flexible Architecture for Implementing Fault Tolerant and Secure Distributed Applications"* (Jean-Charles Fabre).

Isaac Newton Institute (Cambridge University, GB), programme *"Computer Security, Cryptography and Coding Theory"*, présentation de deux communications *"Quantitative Evaluation of Operational Security"* et *Fault Tolerance and Security"* (Yves Deswarte).

6.4 Animations scientifiques

Yves Deswarte a été membre du Comité de Programme de la 2ème *European Dependable Computing Conference (EDCC-2)* qui s'est tenue du 2 au 4 octobre 1996 a Taormina (IT).

Yves Deswarte a été membre du Comité de Programme de la 16ème International Conference on Distributed Computing Systems (ICDCS-16) qui s'est tenu à Hong Kong du 27 au 30 mai 1996.

Yves Deswarte a été membre du Comité de Programme de l'IEEE International Symposium on Security and Privacy qui s'est tenu du 6 au 8 mai 1996 à Oakland (Californie, US), et il sera membre du prochain qui se tiendra du 4 au 7 mai 1997.

Yves Deswarte est membre du Comité de Programme des journées *"Conception de Systèmes adaptifs et spécialisables"*, Rennes, 2-3 avril 1997.

Jean-Charles Fabre a été membre du comité de programme de PDSE'97, *"Second workshop on Software Engineering for Parallel and Distributed Systems"*, qui se tiendra à Boston (US) du 17 au 18 Mai 1997.

Yves Deswarte est membre du Comité de Programme de la conférence IFIP *"Communication and Multimedia Security"* (CMS'97) qui se tiendra du 22 au 23 septembre 1997 à Athènes (GR).

7 Publications

Livres et monographies

[354] J.-C. LAPRIE, J. ARLAT, J.-P. BLANQUART, A. COSTES, Y. CROUZET, Y. DESWARTE, J.-C. FABRE, H. GUILLERMAIN, M. KAÂNICHE, K. KANOUN, C. MAZET, D. POWELL, C. RABÉJAC, P. THÉVENOD, *Guide de la sûreté de fonctionnement*, Cépaduès Éditions, ISBN 2-85428-382-1, 2^{eme} édition, 1996, 380p.

Thèses

[355] V. NICOMETTE, *La protection dans les systèmes à objets répartis*, Thèse de doctorat, Institut National Polytechnique, Toulouse, 17 décembre 1996, 145p., Jury : J.-C. Laprie, Président — Examinateurs : Daniel P. Siewiorek, Michel Raynal, Bruno d'Ausbourg, Yves Deswarte, Jean-Charles Fabre, Michel Riguidel, Jean-Marc Tosques.

Articles et chapitres de livre

[356] J.-C. FABRE, Y. DESWARTE, L. BLAIN, «Tolérance aux fautes et sécurité par fragmentation-redondance-dissémination», *Technique et Science Informatiques*, 1996.

[357] J.-C. FABRE, «Systèmes sûrs de fonctionnement : tolérance aux fautes par protocole à méta-objets», *L'objet*, 1996.

Communications à des congrès, colloques, etc.

[358] M. DACIER, Y. DESWARTE, M. KAANICHE, «Models and tools for quantitative assessment of operational security», *in : 12th International Information Security Conference (IFIP SEC96)*, p. 177–186, Samos (GR), 21–24 mai 1996.

[359] J.-C. FABRE, T. PERENNOU, «FRIENDS. A flexible architecture for implementing fault tolerant and secure distributed applications», *in : 2nd European Dependable Computing Conference (EDCC'2)*, p. 3–19, Taormina (IT), 2–4 octobre 1996. Lecture Notes in Computer Science n°1150, Springer-Verlag.

[360] V. NICOMETTE, Y. DESWARTE, «Symbolic rights and vouchers for access control in distributed object systems», *in : 2nd Asian Computing Science Conference (ASIAN'96)*, p. 192–203, sg, 2–5 décembre 1996. Concurrency and Parallelism, Programming, Networking and Security, Lecture Notes in Computer Science n°1179.

[361] V. NICOMETTE, Y. DESWARTE, «A multilevel security model for distributed object systems», *in : 4th European Symposium on Research in Computer Security (ESORICS'96)*, p. 80–98, Rome (IT), 25–27 septembre 1996. Lecture Notes in Computer Science n°1146.

[362] T. PERENNOU, J. FABRE, «Design and implementation of replication strategies using a metaobjet protocol», *in : Workshop OOPSLA'95 - Reliability and Stability in Distributed Object Systems*, Austin (US), 15–19 novembre 1995.

Rapports de recherche et publications internes

[363] F. CABROLIÉ, «Implantation sur Unix et Chorus de services répartis tolérant les fautes physiques et les intrusions», *Rapport de stage ingénieur cnam*, LAAS-CNRS, mars 1996, 98p.

[364] J.-C. FABRE, T. PÉRENNOU, «Processing of confidential information in distributed systems by fragmentation», *Rapport de recherche n°96390*, LAAS-CNRS, octobre 1996, 21p.

[365] J. LAFFONT, «Editeur de politiques de sécurité utilisant le formalisme des logiques modales», *Rapport de dea*, LAAS-CNRS, 1996, 42p.

[366] F. MELLE, «Utilisation d'un protocole à métaobjet pour le développement de mécanismes de tolérance aux fautes et de sécurité», *Rapport de stage ingénieur iie*, LAAS-CNRS, juillet 1996, 100p.

[367] V. NICOMETTE, Y. DESWARTE, «An Authorization Scheme For Distributed Object Systems», *Rapport de recherche n°96380*, LAAS-CNRS, décembre 1996, 19p.

[368] R. ORTALO, Y. DESWARTE, M. KAANICHE, «Experimenting quantitative evaluation tools for monitoring operational security», *Rapport de recherche n°96369*, LAAS-CNRS, septembre 1996, 21p.

[369] R. ORTALO, «Using deontic logic for security policy specification», *Rapport de recherche n°96380*, LAAS-CNRS, octobre 1996, 14p.

[370] T. PERENNOU, J.-C. FABRE, «Object-Oriented Development of Metaobjects for Distributed Fault Tolerance», *Rapport de recherche n°96476*, LAAS-CNRS, décembre 1996, 20p.

8 Abstract

The SATURNE project aims at a global approach to the tolerance of accidental faults and intentional interaction faults (i.e., intrusions) in distributed systems. This approach enables to tackle reliability and security problems 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" [356] and is now adopted by various academic and industry research teams. 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 to scatter these fragments throughout the distributed system. FRS has been applied to a distributed file storage server, to a distributed security server and more recently to improving the reliability of confidential information processing.

This study lead us to look at the implementation of non-functional characteristics in object-oriented systems, in particular with meta-object protocols (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).