Cartes à puce et méthodes formelles, une lente intégration …

Jean-Louis LANET

Gemplus Research Laboratory, Av du Pic de Bertagne,
13881 Gémenos cedex BP 100.
jean-louis.lanet@gemplus.com

1. Introduction

Les premiers systèmes d'exploitation des cartes à puce ne permettaient pas de télécharger des applications après émission de la carte (post-issuance). Celles-ci étaient donc développées par des programmeurs ayant de fortes compétences en programmation dans les langages de bas niveau. Cette approche, associée à un processus qualité performant, garantissait un niveau de sécurité élevé. Cependant, l'apparition de nouvelles approches, comme les systèmes d'exploitation ouverts tels Java Card [Sun-99] ou Windows for Smart Card [Mic-99], a changé les façons de concevoir non seulement les applications mais aussi les logiciels de base. Avec de tels systèmes d'exploitation il est désormais possible de télécharger du code applicatif dans la carte, en dehors de l'environnement de sécurité du fabricant de carte à puce. Cependant, si cette possibilité ouvre de nouveaux débouchés pour les applications carte à puce, elle soulève de nouveaux challenges en terme de sécurité. Le chargement de codes dans la carte, dont la source est inconnue, implique que la carte doive se protéger sans aide extérieure contre l'arrivée de codes malicieux. L'approche qualité traditionnelle montre là ses limites, car la moindre faute d'implémentation laisse potentiellement une porte ouverte à une attaque. Depuis quelques années, Gemplus s'intéresse au sein du laboratoire de recherche à l'amélioration de la qualité par l'utilisation de techniques formelles. Ces techniques sont appliquées sur le développement, la certification, le test ou des problèmes spécifiques de sécurité comme la recherche de flux illégaux. Pour ce faire nous avons étudié et utilisé différentes technologies : vérifieur de modèles et prouveur de théorèmes ainsi que différents langages comme B, COQ, SMV, SDL …

2. Limites des approches actuelles

Les cartes à puce ne sont pas que de simples objets de stockage, ce sont des ordinateurs aux capacités réduites offrant un niveau de sécurité élevé. Elles incorporent des fonctions de cryptographie comme le triple DES, RSA, AES ou des algorithmes de courbes elliptiques. Elles permettent de construire des protocoles élaborés, en vue de protéger les données de la carte ainsi que les communications. La carte à puce est donc un sous-élément de sécurité d'un système plus vaste et sa défaillance peut entraîner l'obtention illégale de biens ou de services. Rendre les cartes plus sûres nécessite de travailler dans différentes directions (matériel, cryptographie, qualité, méthodologie…) dont deux d'entre elles peuvent tirer parti des techniques formelles : le processus de développement et de test d'une part et le processus de certification d'autre part. Pour le développement il est certain que la complexité des systèmes ne fait que croître, de même que le nombre de fonctionnalités, au rythme de l'accroissement des capacités mémoire des puces. Pour le processus de certification, plus le niveau visé sera important plus la difficulté en terme de modélisation sera élevée. A partir du niveau cinq dans le schéma des Critères Communs il est nécessaire de mettre en œuvre des techniques formelles. Si l'utilisation de techniques formelles est obligatoire pour certaines facettes de la certification, il est possible de les utiliser pour d'autres facettes, comme l'analyse de vulnérabilité, afin de réduire les coûts de cette certification.

2.1. Limites des approches actuelles de développement

Jusqu'à présent la complexité des systèmes d'exploitation était facilement gérable par les architectes logiciel. Le code était développé de façon monolithique et ne différait guère d'une version à l'autre. L'arrivée de machine virtuelle a profondément modifié cette façon de faire. La sécurité des machines virtuelles est divisée en différents modules, certains n'étant même pas sous la responsabilité du maître d'œuvre (comme par exemple le vérifieur de byte code, puisque ne pouvant être logé dans la carte). Cette complexité des systèmes, associée à une sécurité éclatée et la nécessité d'avoir malgré tout une exécution performante et un niveau de qualité du produit, montre les limites de l'approche traditionnelle de la conception des cartes à puce. Il n'est donc pas surprenant que les premiers modèles formels de carte à puce furent dédiés à la Java Card. Cette intense activité de formalisation exprime bien la difficulté qu'ont les chercheurs et les industriels à se convaincre de la correction de ces spécifications, même pour des systèmes d'exploitation dédiés comme [Gri-00].

2.2. Amélioration de la productivité

Actuellement le processus de test requiert de construire une seconde version du logiciel, et de comparer les suites de tests, écrites manuellement, sur les deux implémentations. Si ce processus permet d'obtenir la qualité attendue, elle est néanmoins très coûteuse car elle représente plus de la moitié du coût d'un projet. Il s'agit là d'améliorer la productivité par l'automatisation du processus de test [Mar-01]. Généralement pour tout développement un modèle UML, plus ou moins précis, est élaboré. L'idée est d'utiliser ces modèles afin qu'ils soient utilisables par des outils basés sur des techniques formelles.

Nous avons donc réalisé une évaluation, dans le cadre du test d'application Java Card, du couplage de deux outils TGV [Fer-96] et UMLAUT [Jèz-98]. L'application est modélisée à l'aide de diagrammes UML. L'outil UMLAUT génère un modèle intermédiaire proche d'un LTS (système de transitions étiquetées) qui peut ensuite être facilement interfacé avec TGV pour produire les suites de tests abstraites. Cette expérimentation, associée au développement d'une méthodologie, a montré des résultats prometteurs. Il reste à résoudre le délicat problème de l'intégration des données, afin d'avoir un outil transférable.

Ce travail s'applique principalement à des développements traditionnels mais pourrait aussi s'appliquer au test de conformité sur les développements formels. Il faudrait alors mettre en corrélation le modèle UML et le développement formel, ce qui engendre d'autres problèmes (cohérence, passage d'un modèle à un autre …)

2.3. Besoin de certification

La différenciation entre chaque fabricant se fait généralement au niveau de la sécurité. Chacun revendique un certain niveau de sécurité, qui sera validé par la soumission d'un produit à un laboratoire indépendant. Ce processus de certification est avant tout un moyen de gagner des parts de marché. Cependant dans certains pays (Allemagne, Hongrie…) un certain niveau de certification est obligatoire, en cas d'utilisation de clé de signature. Le schéma de certification concernant les produits de sécurité est actuellement les Critères Communs.

L'utilisation des méthodes formelles intervient à partir du niveau cinq (EAL5) pour la modélisation de la politique de sécurité. Au niveau le plus élevé (EAL7) il est nécessaire que l'architecture de haut niveau soit décrite de manière formelle. Nous avons développé un outil permettant de faciliter une partie de l'analyse de vulnérabilité pour la certification. Cet outil permet de vérifier automatiquement l'absence de flux illégaux pour une configuration d'applets [Bie-00]. Cet outil transforme les graphes d'appel des applications carte dans la sémantique de SMV et insère des invariants représentant la politique de sécurité. Le vérifieur de modèle soit échoue et fournit un contre exemple à analyser, soit valide le modèle. Cette automatisation permet de gagner du temps et de fournir tous les justificatifs nécessaires à la certification.

Le processus de certification a été le moyen de faire sortir du laboratoire de recherche de Gemplus les techniques formelles et semi-formelles comme B et SDL, et de les mettre entre les mains des équipes de développement. Il est à noter que c'est le caractère obligatoire qui a permis cette utilisation et non un critère d'amélioration de la qualité ou de la productivité. Mais c'est un premier pas qui a montré que cette technologie n'était pas une affaire de spécialiste et que tout bon ingénieur pouvait, sinon spécifier, du moins relire des modèles formels.

3. Contraintes industrielles

Si l'utilisation de techniques formelles dans le milieu académique est bien répandue, il n'en est pas de même dans l'industrie (sauf peut-être dans le domaine de la vérification de circuits). Rares sont les industries pour lesquelles l'utilisation de techniques formelles est intégrée dans leur processus de développement. Au mieux ces techniques restent dans des laboratoires d'études avancées, au pire elles ne font que passer. Le contexte de la carte à puce est différent de celui que l'on rencontre généralement. La défaillance d'une carte à puce n'a pas de conséquences catastrophiques pour l'individu, contrairement aux systèmes de transport ou d'énergie. Par contre les pertes financières peuvent être très importantes. S'il existe de bonnes raisons d'utiliser ces méthodes en dehors du processus de certification, de nombreux points doivent être résolus avant de transférer ce savoir-faire. Les améliorations doivent porter principalement sur les outils, les métriques et la méthodologie.

Si aujourd'hui nous sommes convaincus que ces méthodes sont techniquement applicables à la carte à puce, modulo l'adaptation des outils, il n'est absolument pas évident qu'elles seraient économiquement intéressantes. Il faut se rappeler qu'aujourd'hui les attaques contre les carte à puce sont essentiellement des attaques physiques (attaques en courant, en temps, électromagnétisme…) et que les contre mesures actuelles sont essentiellement liées à ce type d'attaque. Les méthodes formelles

pourront résoudre les attaques logiques contre des fautes d'implémentation. Il faut donc convaincre que le surcoût est acceptable. Ce surcoût se mesure en temps de développement mais aussi, et principalement, en place mémoire. Cette dernière est une ressource très critique. Or, si on veut que le surcoût en temps de développement soit compensé par une absence de test unitaire, il faut pouvoir générer de façon sûre le code exécutable. Malheureusement jusqu'à présent aucun outil n'est capable de fournir un code suffisamment optimisé pour répondre aux contraintes drastiques de la carte. De plus il n'existe aucune étude publique, précise et chiffrée, sur l'utilisation de ces techniques dans un domaine proche de la carte.

Notre approche fut de combler une à une ces lacunes par des partenariats adéquats. Ainsi nous menons un projet sur la génération de code C optimisé pour la carte à puce, dans le cadre du projet RNTL BOM [Bom-00]. Nous menons un travail sur la méthodologie et la collecte de métriques dans le projet européen MATISSE [Mat-00] où nous proposons un double développement formel et traditionnel. Dans ce cadre, nous proposons d'intégrer dans une carte à puce un module complet et de comparer les deux développements. Le projet européen VERIFICARD [Ver-00] est pour nous le moyen d'amener la preuve de correction d'un sous-ensemble de la machine virtuelle Java Card et de développer des méthodes d'analyse pour les applets à charger dans la carte. Enfin dans le projet RNTL COTE [Cot-01] nous espérons pouvoir intégrer progressivement le traitement des données pour la génération de suite de test.

4. Conclusions

Le processus de certification est pour nous la partie visible de l'iceberg. Les principaux gains viendront du développement et / ou du test. Il est quasi certain que le test est le sujet qui intéresse le plus les industriels. En effet la phase de test obère le coût global de développement de manière très significative et les responsables sont très sensibles aux gains de productivité.

Après avoir identifié un certain nombre de points faibles dans notre approche nous avons proposé différents projets de recherche. Nous espérons dans un avenir proche pouvoir amener la démonstration qu'il est possible d'intégrer un module entièrement développé à l'aide d'une technique formelle et ce, jusqu'à la génération de code à un coût acceptable.

Les cartes à puce représentent certainement un domaine d'application quasi idéal pour les méthodes formelles. Cependant pour quitter le laboratoire de recherche, un certain nombre de travaux doivent être accomplis. Ceci passe par des partenariats avec les chercheurs du monde académique à travers des projets communs ainsi qu'une meilleure intégration des techniques formelles avec des techniques semi-formelles.

Bibliographie

[Bie-00] P. Bieber, J. Cazin , P. Girard, J.-L. Lanet, V. Wiels , G. Zanon, *Checking Secure Interactions of Smart Card Applets*, ESORICS 2000, pp.1-18, October 2000, Toulouse.

[Bom-00] Projet BOM, http://lifc.univ-comte.fr/PEOPLE/tatibouet/BOM

[Cot-01] Projet COTE , http://www.irisa.fr/cote

[Fer-96] J.-C. Fernandez, C. Jard, T. Jéron, *Using on-the-fly Verification Techniques for the Generation of Test Suites*, CAV'96, New Brunswick, USA, LNCS 1102, Springer, August 96

[Gri-00] Gilles Grimaud, *Camille : un système d'exploitation ouvert pour carte à puce*, Thèse de l'université de Lille, décembre 2000.

[Jèz-98] J.-M. Jèzequel, A. Le Guennec, F. Pennaneac'h, *Validating Distributed Software Modelled with UML*, UML'98, pp. 331-340., 1998

[Mar-01] Hugues Martin, *Une méthodologie de génération automatique de suites de tests pour applets Java Card*, Thèse de l'université de Lille, mars 2001.

[Mat-00] Projet MATISSE, http://www.matisse.dera.gov.uk/

[Mic-99] Microsoft Corp. "Smart Card for Windows" Web site. http://www.microsoft.com/windowsce/smartcard/

[Sun-99] Sun Microsystems, Inc. *Java Card 2.1 Virtual Machine, Run Time Environment, and Application Programming Interface Specification*, Public Review ed., Feb. 1999. http://java.sun.com/products/javacard/javacard21.html

[Ver-00] Projet VERIFICARD, http://www.verificard.com