Article extrait de la REE 2014-5

Améliorer la sécurité et la sûreté de fonctionnement par l'ingénierie de système dirigée par les modèles

Par Ludovic Apvrille[1], Yves Roudier[2]
Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI[1], EURECOM[2]

ABSTRACT *Today's communicating, embedded, and cyber-physical systems consist in an ever-more complex and ubiquitous landscape. Designing safe and secure systems has become a daunting task with respect to the advanced persistent threats they face. We discuss the use of model based system engineering in this context, which we illustrate with the SysML-Sec environment, and the open source software TTool that supports its application. This environment features a unifying approach based on the SysML modeling language. We discuss the methodology that must be used in order to elicit appropriate security and safety requirements and to validate the derived safety and security mechanisms introduced at system level. We illustrate the interest of the joint assessment of security and safety properties based on a use case featuring a communicating automotive system.*

Introduction

Les objets embarqués communicants envahissent progressivement notre quotidien et font ainsi évoluer le périmètre traditionnellement considéré pour les systèmes d'information. De même, on assiste au déploiement de nombreux systèmes cyber-physiques (CPS) dans le domaine industriel. De par leurs capacités de communication avec d'autres objets ou systèmes proches ou avec des services distants, ces cibles potentielles sont de plus en plus exposées à des attaques. Pour ne donner que quelques exemples de la grande diversité des systèmes ayant déjà enduré des attaques réelles, on peut citer pêle-mêle systèmes SCADA (notamment avec le ver Stuxnet), systèmes avioniques ou automobiles, routeurs ADSL, téléphones mobiles, appareils domestiques, ou bien encore consoles de jeu. L'aspect critique de certains de ces systèmes (notamment les moyens de transports et les systèmes industriels) peut bien entendu être mis à profit par un attaquant.

La complexité de ces différents systèmes, notamment en termes de taille, de distribution ou d'hétérogénéité, est un facteur prépondérant en ce qui concerne les risques qu'ils subissent. Cette complexité nécessite l'intégration des études de sécurité au plus tôt dans le cycle de développement logiciel et matériel, afin notamment de déterminer les attaques susceptibles d'impacter la sûreté de fonctionnement et d'y porter remède.

Des attaques problématiques

La réalisation d'attaques repose en général soit sur l'exploitation de vulnérabilités de bas niveau (par exemple, des dépassements de buffer) soit sur celle d'erreurs de conception. Les vulnérabilités de bas niveau peuvent être souvent évitées par de bonnes pratiques de programmation, par l'emploi d'outils d'analyse de code, ou par des tests de sécurité (fuzzing, notamment). Au contraire, les erreurs de conception proviennent plus souvent de l'organisation logicielle ou matérielle du système, de sa complexité et des mécanismes de sécurité intégrés : ces erreurs sont malheureusement peu ciblées par les outils employés couramment par les experts en sécurité.

Par rapport aux systèmes d'information traditionnels, habituellement visés par des attaques, les systèmes d'information comportant des composantes communicantes et embarquées ainsi que les CPS présentent trois particularités qui en font des cibles bien plus dangereuses :

- Si une attaque exploite une faille matérielle du système, il peut être difficile voire impossible de corriger ("patcher") le système par une simple mise à jour logicielle, notamment en présence de choix architecturaux implantés matériellement (accélérateur matériel, bus mémoire, filtrage réseau). En comparaison, dans le cas d'un système d'information classique, l'isolation, la reconfiguration et la réinitialisation du système suffisent normalement à effectuer sa mise à niveau avec succès ;
- L'application des mises à jour de sécurité est plus complexe pour un système embarqué. Par exemple, la mise à jour d'un véhicule nécessite son rappel en concession. Les utilisateurs sont aussi bien moins enclins à effectuer des mises à jour, comme on l'observe par exemple dans le cas de caméras IP, même quand elles sont accessibles à l'utilisateur. Au contraire la mise à jour d'un ordinateur personnel ou d'un ser-

veur sont d'une part faciles à gérer à distance de manière automatique et peuvent être assurées par le biais de mécanismes de gestion et de contrôle de parc informatique ;

- L'effet d'une attaque est bien plus dévastateur dans le cas d'un système critique et peut même éventuellement mettre en danger la sécurité des personnes. De plus, le déploiement massif des composantes embarquées de systèmes d'information (ordinateurs, téléphones mobiles, objets connectés) signifie qu'un attaquant réussissant à pénétrer l'un de ces dispositifs dispose d'une arme capable d'effectuer des attaques de grande ampleur de type *botnet*, ayant pour effet du déni de service distribué, du pourriel/spam, voire une contribution à des calculs distribués pour mener une attaque cryptographique.

Modéliser un système: le rôle central de l'architecture

Les composants fondamentaux de l'architecture du système considéré doivent être modélisés avant de pouvoir décrire et analyser les attaques potentielles ainsi que le déploiement des mécanismes de sécurité. Dans le cadre des questions de sûreté de fonctionnement, ces composants fondamentaux doivent aussi être analysés afin de mieux cerner les parties critiques du système.

Partitionnement logiciel/matériel

Les systèmes à dominante logicielle sont classiquement conçus en suivant un cycle de développement en V, dans lequel des étapes de « construction » (exigences, analyse, conception, déploiement) sont suivies par des étapes de vérification (simulations, tests, vérifications formelles). Dans le cadre des systèmes embarqués, le cycle de développement du logiciel en V ne peut bien entendu être mis en œuvre qu'une fois que les fonctions devant être implantées de façon logicielle ont été sélectionnées : ainsi, le partitionnement logiciel/matériel doit avoir lieu avant que le cycle en V logiciel ne démarre. Ce partitionnement logiciel/matériel repose en général sur le modèle dit en "Y" [1] qui comporte trois étapes. Les deux premières consistent à élaborer des modèles fonctionnel et architectural du système. La troisième étape consiste à projeter les fonctions ainsi que leurs communications sur les architectures candidates : une fonction projetée sur un processeur sera ainsi une fonction logicielle, alors qu'une fonction projetée sur un accélérateur matériel sera une fonction implantée de façon matérielle.

Dimensionnement

La phase de partitionnement est bien entendu très importante et peut remettre en cause les choix de répartition entre le logiciel et le matériel. Par exemple, un CPU peut être mal dimensionné, et donc incapable d'exécuter une fonction de traitement du signal qui devrait alors être confiée à un circuit spécialisé. Le coût de rétro-ingénierie plus tard dans le cycle de développement peut être alors très important, et en outre retarder la date de mise sur le marché du système.

Dans certains systèmes critiques, les modèles inhérents à l'exploration d'architectures n'ont pas vocation à rechercher les temps d'exécution dans le pire cas (Worst Case Execution Time). Au contraire, les techniques de dimensionnement basées sur le network calculus [2] ont pour objectif le calcul des pires scénarios.

Logique métier et distribution des données

Indépendamment de l'utilisation de composants matériels, les systèmes d'information nécessitent de plus en plus de définir le partitionnement des fonctions de la logique métier ainsi que la distribution des données. L'arrivée des techniques de virtualisation (par exemple au niveau des systèmes d'exploitation, comme par exemple L4 ou TrustedZone) a notamment complètement transformé les principes d'architecture cloisonnée en vigueur dans les systèmes d'information. On constate aussi clairement une évolution dans le cadre des services cloud. Dans ce cas, il s'agit d'empêcher des atteintes aux programmes des utilisateurs dans ces systèmes éminemment partagés par l'introduction de compartiments sécurisés. Cependant de nombreux traitements dépassent le cadre d'un compartiment pour des raisons de parallélisme. On ne peut que constater par exemple que de nombreux traitements, notamment dans le cadre du *BigData,* conduisent à distribuer des données de manière très désordonnée, contrairement aux spécifications des politiques de contrôle d'accès ou d'usage habituellement mises en œuvre dans des systèmes d'information traditionnels. Dans ce cadre également, s'assurer du partitionnement des traitements et de leurs projections sur des serveurs devient extrêmement important.

L'ingénierie système dirigée par les modèles pour maîtriser la complexité

De très nombreuses recherches ont employé l'ingénierie dirigée par les modèles pour la conception de systèmes complexes, distribués et temps-réel, en visant notamment l'analyse de propriétés de sûreté de fonctionnement. Ainsi, ces contributions concernent fréquemment l'analyse des contraintes temps-réel, l'ordonnancement, l'allocation de ressources et la concurrence. Les techniques de simulation et de vérification formelle appliquées sur des spécifications écrites dans des formalismes spécifiques ont largement été adaptées, souvent via des techniques de transformations de modèles, aux modèles de

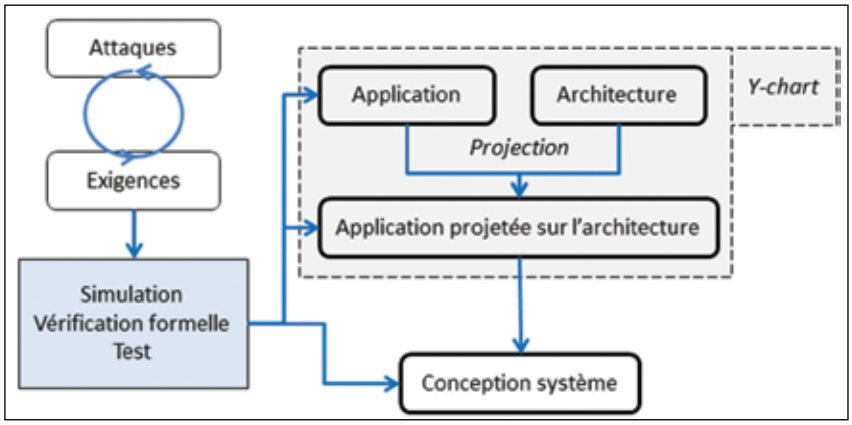

Figure 1 : La méthodologie SysML-Sec.

type UML ou SysML, et implantés dans des ateliers de modélisation graphique comme PolarSys ou Papyrus.

Spécifier les besoins de sécurité d'un système

La sécurité (contrer les malveillances) et la préservation de la vie privée sont à l'inverse de la sûreté de fonctionnement des contraintes peu prises en compte dans la conception des systèmes d'information distribués. Des besoins apparaissent généralement a posteriori, lorsque des failles sont découvertes une fois le système mis en production. Cette constatation revêt un caractère particulièrement dangereux lorsque ces failles de sécurité concernent des aspects critiques de ces systèmes. Un premier problème concerne donc la détermination des objectifs au plus tôt du cycle de conception et développement. Un deuxième problème concerne la compréhension des assurances de sûreté de fonctionnement impactées, aussi bien au niveau des ressources touchées que de leurs interactions.

Objectifs de sécurité et menaces : ingénierie des exigences

De nombreux travaux ont déjà abordé différentes facettes de la modélisation des objectifs et menaces de sécurité par l'ingénierie des exigences dans les systèmes d'information. [3] classifie les techniques de développement d'exigences de sécurité en quatre domaines : les approches orientées objectifs, les approches orientées modèles, les approches orientées processus, et enfin les approches orientées problèmes, les deux premières étant les plus en phase avec le cycle de conception système. Les approches orientées objectifs sont les plus intéressantes pour l'analyse des exigences de sécurité aux stades les plus précoces de la spécification et de conception de l'architecture du système et permettent de mieux analyser les exigences de sécurité nécessaires sans être influencé par une implantation spécifique. Plus proches des mécanismes de sécurité, les environnements qui suivent l'approche orientée modèles sont mieux adaptés en termes de vérification formelle des protocoles cryptographiques, mais sur une architecture déjà plus détaillée tant du point de vue matériel que logiciel.

Une approche unificatrice basée sur SysML

L'environnement SysML-Sec [4] combine approche orientée objectifs pour les exigences et approche orientée modèles pour l'architecture système et les menaces. Il a pour premier objectif de faciliter la collaboration et la communication entre les experts en conception système et les spécialistes en sécurité durant l'ensemble des phases méthodologiques de développement de ces systèmes. C'est ce qui justifie notre recours aux langages UML/SysML, spécifiés par l'OMG et l'INCOSE. Ces langages sont notamment utilisés de plus en plus largement dans la conception système, notamment dans le contexte des systèmes embarqués [5]. Dans notre environnement, les objectifs de sécurité comme les attaques peuvent être décrits intégralement en SysML.

Méthodologie

La méthodologie associée à SysML-Sec est basée sur une première phase d'analyse système basée sur le modèle en «Y» suivie par une deuxième phase de conception du logiciel basée sur le cycle en "V". Comme la méthodologie orientée objectifs TwinPeaks, notre approche met particulièrement l'accent sur des allers-retours fréquents et agiles entre l'architecture et l'identification des ressources à protéger, des menaces qui pèsent sur elles, et des exigences de sécurité ou de sûreté. SysML-Sec intègre de surcroît le modèle en "Y" [1] et ses procédures d'allocation sous-jacentes nécessaires à l'évaluation de la sûreté de fonctionnement et de la performance système. Cette méthodologie est résumée à la figure 1.

La phase d'analyse système consiste à identifier les exigences de sécurité et les attaques conjointement avec l'identification des principales fonctions (*"application"*), ainsi que les architectures candidates et la projection des fonctions sur l'architecture (*"mapping"*). A l'étape fonctionnelle, la simulation ou la vérification formelle ont pour objectif d'identifier des problèmes de sûreté de fonctionnement, typiquement des situations d'interblocage (*"deadlock"*). Les modèles fonctionnels étant non temporels, aucune étude de performance ne peut être réalisée à ce stade. La projection des fonctions sur des architectures matérielles permet d'attribuer un temps logique et/ou physique aux opérateurs de description des fonc-

tions. Les simulations ou vérifications formelles qui sont ainsi réalisées après projection ont pour objectif l'évaluation des performances sur la plate-forme matérielle sélectionnée, typiquement les latences de traitement, les charges des processeurs, les temps de communication et enfin la charge des bus. Bien entendu, ces performances sont impactées d'une part par les aspects logiques des fonctions, d'autre part par les aspects sécurité de ces mêmes fonctions. Par exemple, tel protocole de sécurité génère du trafic supplémentaire sur un bus, ou tel algorithme cryptographique prend un temps de calcul différent selon qu'il est déployé sur un accélérateur matériel ou sur un processeur. L'étude de performance permet alors de sélectionner la plate-forme au regard des contraintes temps-réel (qui sont en général des contraintes liées à la sûreté de fonctionnement) et des contraintes de sécurité.

La phase de conception suit la phase d'analyse. L'architecture logicielle et matérielle étant décidée à la phase précédente, cette phase consiste donc à concevoir les fonctions logicielles de façon plus précise jusqu'à les raffiner le plus exhaustivement possible. Ce raffinement concerne aussi les mécanismes de sécurité implantés de façon logicielle, et en particulier les protocoles cryptographiques. Lors des premiers raffinements, des techniques de simulation de modèle permettent de déboguer ces derniers, alors que la vérification formelle permet, lorsque le modèle n'est pas encore trop riche, de prouver formellement des propriétés de sûreté de fonctionnement (typiquement, la vivacité d'un état, ou son accessibilité), et des propriétés de sécurité (typiquement, la confidentialité d'une donnée ou l'authenticité d'un message). Lorsque le modèle devient trop riche pour être vérifié de façon efficace, il convient alors de générer automatiquement du code exécutable, sur lequel des tests (tests unitaires, tests de sécurité) peuvent être réalisés.

Validations pour la sûreté et la sécurité

L'ingénierie système dirigée par les modèles encourage et favorise naturellement la validation au plus tôt des propriétés du système. SysML-Sec vise à de telles validations soit depuis les modèles de partitionnement, soit depuis les modèles de conception. Ces validations sont implantées dans l'outil TTool[1] (utilisé également pour la modélisation). Elles consistent en la transformation des modèles semi-formels SysML-Sec en des spécifications formelles, en l'analyse des propriétés visées, enfin en la remontée sur les modèles SysML-Sec des résultats de la validation afin de les visualiser.

Modélisation et preuves de propriétés de sûreté de fonctionnement

La validation de la conception concerne la vérification des propriétés de sûreté de fonctionnement. De façon générale, si les modèles des systèmes sont communément réalisés avec des langages graphiques, les langages les plus usités pour exprimer les propriétés formelles de sûreté sont basés sur des langages textuels, notamment LTL/CTL (ou des dérivés). La modélisation en SysML des propriétés de sûreté de fonctionnement repose sur les diagrammes paramétriques afin d'exprimer ces propriétés sous la forme de relations logiques et temporelles. La validation doit souvent s'appuyer sur plusieurs techniques aussi bien statiques, comme la vérification formelle (par exemple avec des modèles d'automates temporisés [6]), que dynamiques, comme la simulation du système [7], ou la génération de code exécutable et son test [8]. L'intégration de ces techniques dans un seul atelier (TTool) facilite le travail d'ingénierie.

Propriétés de sécurité

Les propriétés de sécurité liées à la conception peuvent aussi être vérifiées par preuve formelle. SysML-Sec introduit aussi des extensions au langage graphique afin de permettre la validation des protocoles cryptographiques développés pour sécuriser la communication entre blocs SysML. Les propriétés de sécurité de confidentialité, intégrité et authenticité sont généralement plus simples à exprimer que celles de sûreté. TTool implante une transformation des modèles de conception vers une spécification en pi-calcul ensuite validée dans l'outil ProVerif. Les diagrammes SysML-Sec peuvent être annotés à cet effet avec des propriétés de sécurité (confidentialité, authenticité) [4].

Sûreté et sécurité, sûreté contre sécurité

Le lien entre exigences de sûreté et de sécurité est assez rarement traité au niveau du partitionnement logiciel/matériel. Il nous semble pourtant crucial de réaliser ce lien au plus tôt dans le cycle de développement. En effet, l'introduction de mécanismes supplémentaires liés à la protection du système peut remettre en cause partitionnement, et besoins de performance en calcul et en communications. Eames and Moffet [9], et plus récemment Pièrre-Cambacédès et Bouissou [10] et Raspotnig et Opdahl se sont intéressés eux aussi aux liens entre exigences de sûreté et de sécurité. Par exemple, ces derniers auteurs ont mis au point des modèles pour décrire les conflits entre exigences, mais aussi les « renforcements » entre exigences de natures différentes qui convergent vers les mêmes choix d'architecture. Ces modèles per-

[1] http://ttool.telecom-paristech.fr/

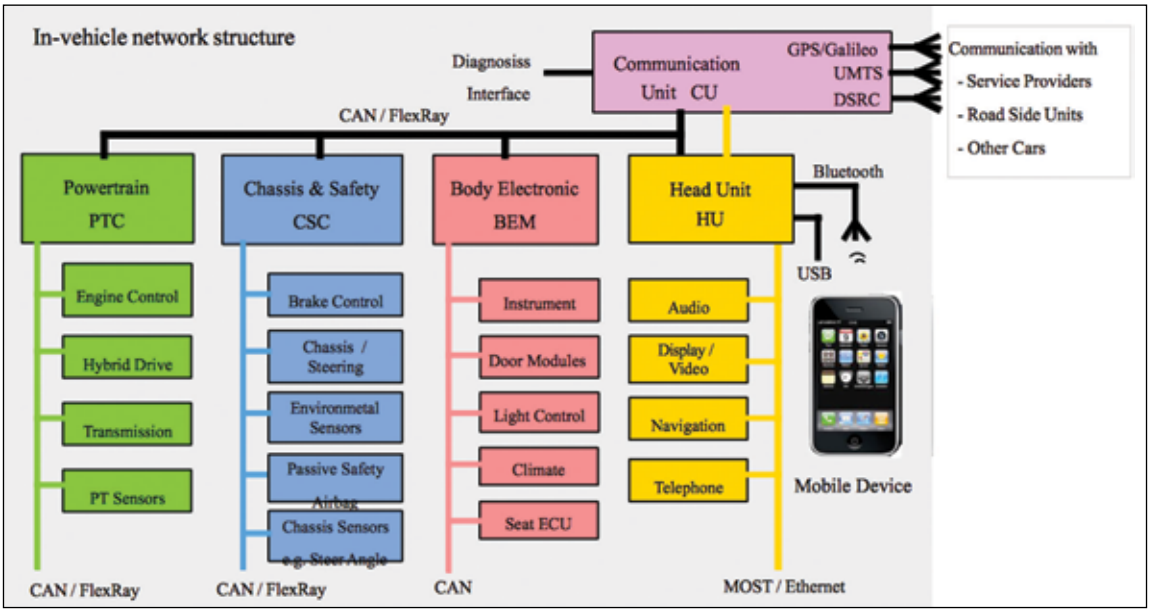

Figure 2 : Architecture de référence EVITA d'un véhicule connecté [12].

mettent aussi de décrire des cas de dépendances conditionnelles.

Comme nous l'illustrons dans la section suivante, SysML-Sec se focalise sur l'évaluation de l'innocuité de mécanismes de sécurité vis-à-vis d'exigences de sûreté, par l'utilisation de vérifications soit au niveau du partitionnement, soit en phase de conception. Au niveau partitionnement, les mécanismes sont évalués en termes d'usage de la plate-forme, comme par exemple la puissance de calcul nécessaire, ou l'utilisation de la bande passante des bus. Le raffinement successif des modèles SysML-Sec permet progressivement une étude plus fine des mécanismes cryptographiques (accessibilité d'un état d'un protocole cryptographique, par exemple). Ces études reposent principalement sur des simulations de charge, par exemple des bus de communication ou des calculateurs, ou de respect des échéances temps-réel, par rapport aux fonctionnalités attendues. En phase de conception, nous employons aussi des mécanismes d'inférences logiques afin de vérifier la cohérence des exigences et la couverture des attaques recensées.

Exemple : véhicule communicant

Dans la suite de l'article, nous illustrons la première partie de la méthodologie SysML-Sec - à savoir, l'identification des exigences et des attaques conjointement avec la construction d'une architecture sûre et sécurisée - sur une étude de cas de système automobile communicant issue du projet de recherche européen FP7 EVITA [12]. EVITA a défini la première architecture de sécurité généraliste pour les systèmes embarqués automobiles communicants. Les architectures embarquées véhiculaires comportent des composantes critiques pour la sûreté alors même qu'elles comprennent à l'heure actuelle une centaine de calculateurs (ou "ECU" pour "Electronic Control Unit") interconnectés avec des bus de type CAN ou FlexRay. Les attaques sur ces systèmes [13] s'expliquent soit par des motifs économiques (activer des options payantes, voler un véhicule), soit par des attaques liées à la malveillance. Les véhicules sont de plus souvent connectés par Internet mobile à des systèmes d'information d'où ils puisent par exemple des données de navigation ou qui leur offrent des mécanismes de mise à jour par communication sans fil des micro-logiciels embarqués. Les recherches actuelles sur les réseaux de communication automobiles (VANETs) vont encore augmenter l'interconnexion des véhicules, aussi bien entre eux qu'avec l'infrastructure routière (péages, feux rouges), ce qui ne fera qu'accroître le nombre de vecteurs d'attaques potentiels.

L'architecture de référence d'un système automobile connecté est présentée à la figure 2. Cette architecture comporte différents domaines (*Powertrain, Chassis & Safety, Communication Unit, Head Unit*, etc.), implantés par des sous-systèmes contenant un ou plusieurs bus et des calculateurs. Un bus principal, souvent de technologie CAN ou FlexRay, interconnecte l'ensemble des calculateurs passerelles contrôlant l'accès aux domaines. De plus, l'unité de communication ("*Communication Unit*") comporte plusieurs interfaces réseaux, notamment vers Internet, et l'ordinateur de bord ("*Head Unit*") permet des connexions Bluetooth et USB.

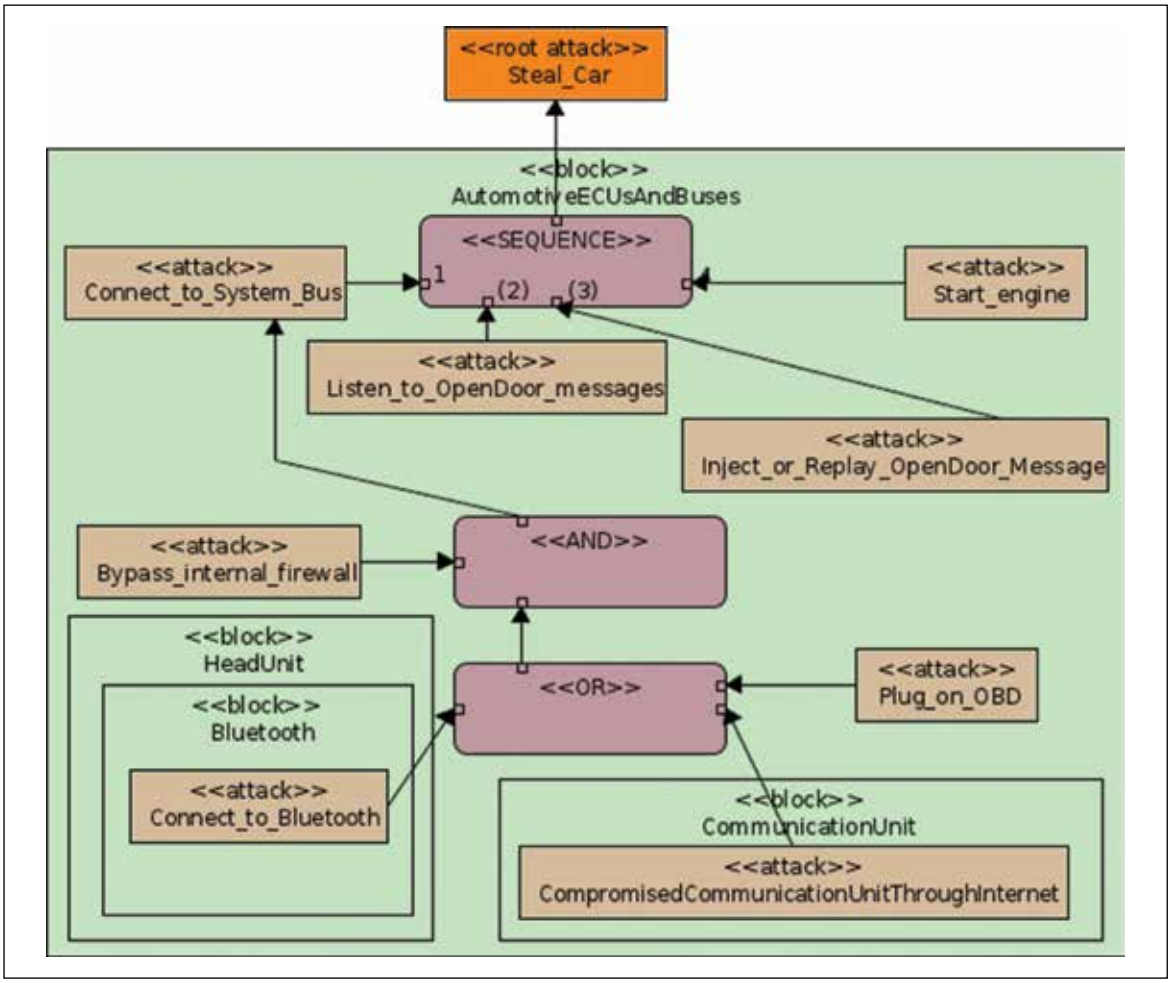

Figure 3 : Description des attaques avec le diagramme paramétrique SysML.

Les modèles : menaces et scénarios d'attaques

Les attaques sont traditionnellement modélisées sous la forme d'arbres d'attaques, d'ailleurs avec de nombreuses variantes. Si de tels arbres permettent de décomposer hiérarchiquement une attaque, ils ne sont pas très flexibles pour décrire des scénarios plus complexes contenant de nombreuses variante. Pourtant, les attaques actuellement conduites sont souvent construites à partir de l'exploitation de plusieurs vulnérabilités, dans un ordre parfois important, voire avec des contraintes temporelles entre différentes phases. C'est par exemple le cas typique de l'exploitation d'une transaction bancaire effectuée à l'aide d'un jeton d'authentification comportant une date d'expiration. Ces considérations nous ont amenés à proposer un formalisme plus étendu que les arbres d'attaques, et à l'intégrer dans SysML-Sec : les équations d'attaques.

La figure 4 illustre cette représentation sur un exemple très simplifié tiré de notre étude de cas. Une ressource centrale (*AutomotiveECUsandBuses*) est la cible d'un attaquant qui désire voler le véhicule (Attaque principale *Steal_Car*). Pour ce faire, il doit d'abord se connecter sur le bus système afin, par la suite, d'ouvrir les portes (cette attaque est optionnelle, car un voleur pourrait décider de casser une vitre), et enfin de démarrer le moteur. L'accès sur le bus système peut se faire en se connectant soit via une interface extérieure (Bluetooth ou Internet), soit par un connecteur de diagnostic dans le véhicule (prise OBD-II). Dans tous les cas, il faut contourner le pare-feu. Le diagramme explicite les attaques en les positionnant vis-à-vis des ressources et éléments d'architecture (les blocs SysML). Ces attaques peuvent être reliées aux exigences de sécurité exprimées sur un autre diagramme déjà employé en SysML pour exprimer les exigences fonctionnelles.

Mécanismes de sécurité et analyse de la satisfaction des exigences de sûreté de fonctionnement

Les diagrammes SysML permettent de modéliser d'une part l'aspect fonctionnel de l'application, et d'autre part l'architecture matérielle et les projections de fonctions sur cette architecture matérielle.

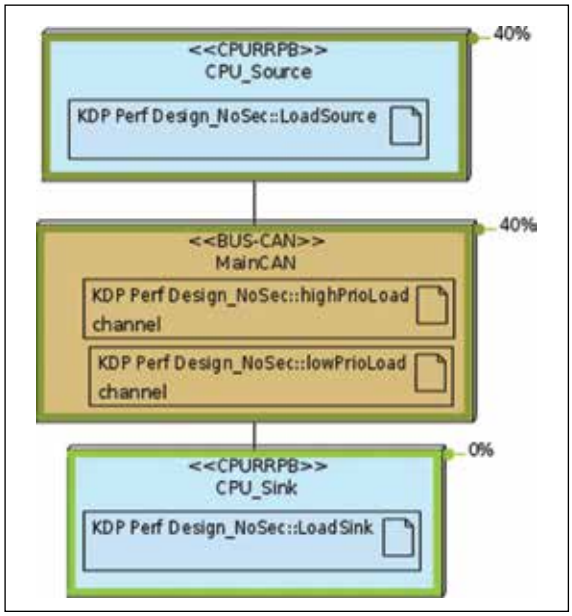

Figure 4 : Simulation de l'architecture automobile sans la distribution de clés.

Du point de vue fonctionnel, des ports et liens entre blocs permettent d'exprimer les échanges d'information (événements de synchronisation et échanges de données). Afin de faciliter la modélisation et les performances en simulation et vérification formelle, la modélisation ne porte que sur la quantité de données échangées, c'est-à-dire que les valeurs de ces données ne sont pas prises en comptes pour évaluer la performance, la sûreté de fonctionnement ou la sécurité.

La projection des tâches et des communications de niveau fonctionnel fait appel à des artefacts sur les nœuds, ces derniers représentant les ressources à protéger et qui peuvent aussi être représentées dans les équations d'attaques.

L'étude de cas concerne l'ajout de mécanismes de sécurité à l'architecture de référence. L'un de ces mécanismes consiste en la distribution de clés cryptographiques de session entre domaines et sous-domaines. Un calculateur *ECU1* s'adresse au serveur de clé (*KeyMaster*) pour lui demander la génération d'une clé de groupe afin de communiquer de façon sécurisée avec d'autres calculateurs (*ECUN*).

Une première simulation est réalisée sur l'architecture automobile (figure 5). L'idée principale est d'évaluer avant tout la charge sur le bus CAN principal, car il achemine, en particulier, les messages urgents entre sous-domaines. Ainsi, les différents trafics de données sont modélisés par la génération de paquets urgents et non urgents sur le bus, qui revient à une charge d'environ 40 %.

Dans un deuxième temps, nous souhaitons étudier l'impact en performance et en sûreté de fonctionnement de l'ajout de la fonction de distribution de clés. Pour ce faire, un sous-ensemble plus important de l'architecture est modélisé : notamment, les accélérateurs cryptographiques *(HSM - Hardware Security Module*), le calculateur à l'origine de la génération de clé (*ECU1*), le gestionnaire principal de clé (*KM*), et le ou les calculateurs qui feront partie du groupe de la clé de session. La simulation met ainsi en évidence, notamment par un marquage sur les diagrammes d'architecture (bus colorés en rouge de la figure 6 et taux d'utilisation associés) une charge de trafic très importante sur le bus principal lors des distributions de clés (dégradation en performance). La simulation permet aussi de détecter une latence augmentée pour toutes les classes de trafic : les choix de sécurité impactent ainsi la sûreté de fonctionnement. L'authentification doit donc être distribuée sur plusieurs messages successifs pour éviter tout problème et la distribution de clés doit être échelonnée dans le temps. Plus d'informations sur cette étude sont disponibles dans [14].

Conclusions et perspectives

Ces dernières années ont été marquées par la recrudescence des attaques sur les systèmes embarqués, systèmes cyber-physiques, et sur les systèmes d'information à composantes communicantes et embarquées en raison de leur interconnexion grandissante. Garantir un temps de mise sur le marché court tout en offrant des assurances de sûreté de fonctionnement et de sécurité nécessite de revoir les méthodologies et outils de conception et de validation issus des systèmes d'information traditionnels. L'adoption grandissante de l'ingénierie des modèles pour la conception de ces systèmes, de plus en plus complexes, nécessite par ailleurs le développement de modèles de sûreté de fonctionnement et de sécurité adaptés pour la production de systèmes de qualité industrielle. Ces modèles doivent permettre de capturer objectifs de sécurité et attaques, tout comme échéances critiques et besoins en bande passante et en puissance de calcul au niveau système.

Nous proposons ainsi un support nommé SysML-Sec et basé sur un langage graphique populaire parmi les experts système et bénéficiant d'un outil support libre, TTool. SysML-Sec vise ainsi à réaliser des analyses de sécurité orientées modèles s'appuyant sur des techniques de simulation comme de vérification formelle permettant de valider les choix de conception au plus tôt, et notamment d'évaluer l'impact de la

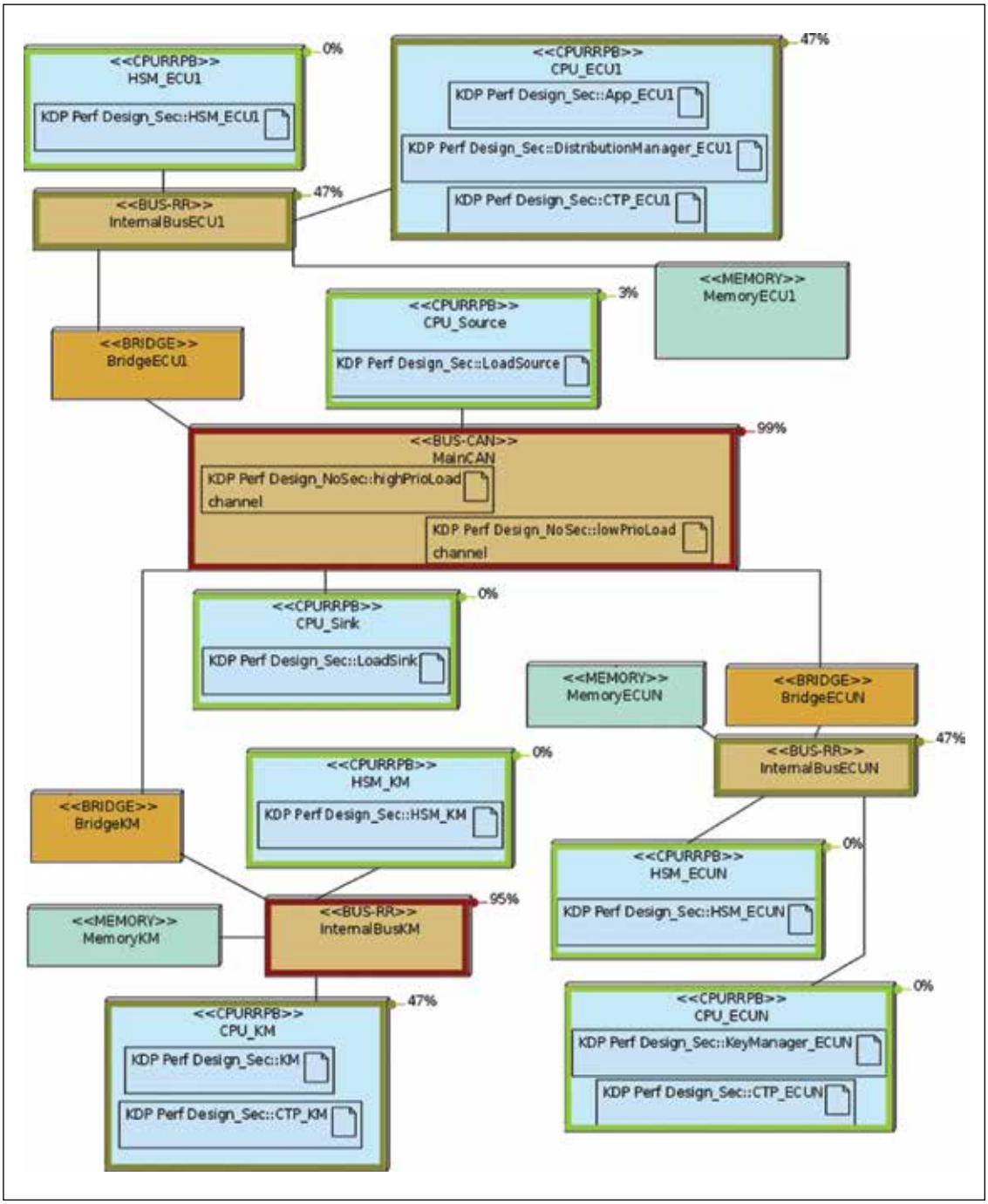

Figure 5 : Simulation de l'architecture automobile avec la distribution de clés.

sécurité sur d'autres aspects, comme la sûreté et la performance générale, mais aussi l'architecture système.

Notre approche a été définie et mise en œuvre dans le cadre de la sécurisation d'une architecture embarquée automobile. L'étude de cas présentée dans cet article est issue de ce travail, et montre l'intérêt d'intégrer étroitement étude et conception des mécanismes de sécurité, des attaques et du partitionnement système.

Le raisonnement sur les modèles, notamment en phase de conception, est un autre objectif important de nos travaux. Nous visons notamment à vérifier que des fonctionnalités notamment critiques ne sont pas inhibées par des mécanismes de sécurité comme le chiffrement ou de filtrage réseau. Raisonner sur les modèles peut être mis en œuvre par le biais de règles d'inférence logiques. Nous travaillons à la mise au point d'une extension des mécanismes déjà employés pour assurer traçabilité des exigences de sécurité et vérification de la couverture des attaques.

Références

[1] F. Balarin, Y. Watanabe, H . Hsieh., L. Lavagno, C. Passerone & A. Sangiovanni-Vincentelli, (avril 2003), "*Metropolis: An Integrated Electronic System Design Environment*", Computer, vol. 36, no 4, p. 45-52.

[2] J. Leboudec, P. Thiran, (2001), "*Network Calculus*", Springer Verlag LNCS volume 2050.

[3] A. Nhlabatsi, B. Nuseibeh & Y. Yu, (2010), "*Security Requirements Engineering for Evolving Software Systems: a survey*", Rapport technique no 1, vol. 1. The Open University.

[4] L. Apvrille, Y. Roudier, (septembre 2013), "SysML-Sec: A SysML Environment for the Design and Development of Secure Embedded Systems", In APCOSEC 2013. Yokohama, Japan.

[5] F. Kordon, J. Hugues, A. Canals & A. Dohet, (2013), "*Embedded Systems: Analysis and Modeling with SysML, UML, and AADL*", France, Lavoisier, Hermès Sciences.

[6] J. Bengtsson, W. Yi, (2004), "*Timed Automata: Semantics, Algorithms and Tools*", In Lecture notes on concurrency and petri nets, p. 87-124. W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag.

[7] L. Apvrille, P. de Saqui-Sannes , (2011, septembre), « *AVATAR/TTool : un environnement en mode libre pour SysML temps réel* », Génie Logiciel, no 98, p. 22–26.

[8] L. Apvrille, A. Becoulet, (February 2012), "*Prototyping an Embedded Automotive System from its UML/SysML Models*". in ERTSS'2012. Toulouse, France.

[9] D. P. Eames, J. Moffett, (1999), "*The Integration of Safety and Security Requirements*", in Safecomp, p. 468-480.

[10] L. Pietre-Cambacedes, M. Bouissou, (2013), "*Cross-Fertilization between Safety and Security Engineering*", Rel. Eng. & Sys. Safety, vol. 110, p. 110-126.

[11] C. Raspotnig, A. L. Opdahl, (2013), "*Comparing Risk Identification Techniques for Safety and Security Requirements*", Journal of Systems and Software, vol. 86, no 4, p. 1124-1151.

[12] E. Kelling, M. Friedewald, T. Leimbach, M. Menzel, P. Säger, H. Seudié & al., (2009), "*Specification and Evaluation of e-Security Relevant Use cases*", Deliverable D2.1. EVITA Project.

[13] T. K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway & *al.*, "*Experimental security analysis of a modern automobile*," in 31st Security and Privacy, IEEE, 2010.

[14] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille & D Scheuermann, (September 2011,), "*C2X Communication: Securing the Last Meter*", in The 4th IEEE International Symposium on Wireless Vehicular Communications: WIVEC 2011. San-Francisco, USA.

LES AUTEURS

Ludovic Apvrille est enseignant-chercheur dans le département Communications et Électronique de Télécom ParisTech, après un doctorat en informatique effectué à l'ENSICA/ISAE, en collaboration avec Thalès Alenia Space et le LAAS, et un post-doctorat effectué à l'Université Concordia (Canada). Il a obtenu son habilitation à diriger les recherches en 2012. Sa thématique de recherche concerne l'analyse et la conception de systèmes embarqués sûrs et sécurisés, et la définition des outils associés. Il applique ses travaux à des secteurs tels que les terminaux mobiles et les transports. Enfin, il est le principal contributeur de TTool, un outil de modélisation et de vérification UML/SysML dédié aux systèmes embarqués.

Yves Roudier est maître de conférences au département Réseaux et Sécurité d'EURECOM depuis 1998, après un doctorat d'informatique à l'Université de Nice Sophia Antipolis puis deux ans de post-doctorat à l'Electrotechnical Laboratory (ETL) à Tsukuba (Japon). Il s'intéresse à l'ingénierie système et logicielle pour la sécurité, et notamment aux techniques d'ingénierie des modèles et de programmation par aspects, tout comme à la cryptographie appliquée, entre autres dans les domaines des véhicules communicants, de l'informatique en nuage ou de l'informatique diffuse.