13e Congrés des Doctorants 2012
EDSYS (École Doctorale Systèmes)
9 et 10 Fevrier, Toulouse

Un routeur IP embarqué et sécurisé pour l'aéronautique

Antoine Varet (avaret@recherche.enac.fr)
LEOPART/CNS/ENAC, Toulouse
Ecole Nationale de l'Aviation Civile, Toulouse
Thèse encadrée par : Nicolas Larrieu (ENAC/LEOPART) et Christophe Macabiau (ENAC/LTSTA)

Résumé *L'importance des réseaux de télécommunication dans l'aéronautique croît au fil des générations avioniques. Ainsi, les communications entre pilotes et contrôleurs, jusqu'à récemment exclusivement analogiques, sont désormais renforcées par des données numériques. Afin de permettre de nouvelles interactions sûres et sécurisées entre les systèmes avioniques, nous élaborons un routeur IP appelé "routeur Sécurisé Nouvelle Génération (SNG)". Cet article aborde différents aspects du développement de ce nœud central des réseaux informatiques, nœud adapté ici à l'aéronautique. Dans un premier temps sera présentée la méthodologie de développement élaborée pour le routeur puis son application. Une discussion relative à la sécurisation des flux de données concluera cet article.*

A. INTRODUCTION

L'accroissement de performances des systèmes aéronautiques permet aujourd'hui d'envisager l'utilisation de nouvelles technologies dans le cadre des systèmes aéronautiques embarqués à bord des avions, ainsi que l'ouverture de réseaux avioniques, jusqu'alors restés fermés pour garantir leur sûreté et leur sécurité, aux réseaux publics tels que le réseau Internet. Ces nouvelles technologies s'accompagnent donc de nouveaux besoins en terme de sécurité tout en maintenant des exigences élevées de sûreté de fonctionnement.

C'est pourquoi l'entreprise Thalès Avionique, fournisseur mondialement reconnu d'équipements et de services dans le domaine avionique, supporte de nombreux projets novateurs dans ce domaine. Nous travaillons sur la création d'un routeur Sécurisé Nouvelle Génération (Routeur SNG). Il permet de multiplexer, par exemple, des flux critiques (tels que les communications pilotes/contrôleurs) avec des flux non critiques (streaming pour les passagers) sur un même canal de communication, en garantissant la sûreté et la sécurité des communications.

Notre travail s'inscrit dans le cadre d'une thèse co-financée par l'Ecole Nationale de l'Aviation Civile et l'entreprise Thalès Avionique Toulouse.

1. Besoin d'un routeur IP embarqué

Les premières générations de systèmes avioniques logiciels étaient basées sur des relations directes entre systèmes : lorsqu'un capteur transmettait une information à deux ordinateurs de bord, les données étaient dupliquées sur deux canaux de communication indépendants, chacun étant dédié à un seul récepteur. L'arrivée de nouvelles technologies a permis de fournir de nouveaux services aux équipages et a introduit de nouvelles interactions. Par exemple, les systèmes Integrated Modular Avionics (IMA[7]) permettent l'exécution de plusieurs logiciels indépendants au sein d'un même module matériel, évitant ainsi de dupliquer les connexions physiques (un seul bus destiné au matériel transporte les données multiplexées pour les différentes applications logicielles). Le nombre croissant de systèmes avioniques a cependant maintenu un besoin élevé d'interconnexions entre systèmes, chacune ayant des contraintes potentiellement différentes.

Le développement de l'Airbus A380 a augmenté la scalabilité [1] des communications entre systèmes par l'introduction de l'Avionics Full-DupleX switched ethernet (AFDX[11]). L'AFDX est un protocole réseau de niveau liaison (couche 2 "liaison" du modèle Open Systems Interconnection, OSI) basé sur le protocole Ethernet, fiabilisé et redondant, permettant la transmission générique de données entre systèmes émetteurs et récepteurs. Ainsi, les systèmes AFDX sont connectés à l'aide d'une interface Ethernet standardisée. Les données sont encapsulées dans des paquets IP (Internet Protocol), eux-mêmes transmis dans des trames Ethernet. Des commutateurs AFDX assurent la direction des trames au sein du réseau AFDX.

Cependant, la taille de la topologie d'un réseau AFDX est limitée techniquement : le réseau fermé actuel de l'A380 contient une centaine de nœuds ter-

1. Anglicisme signifiant la capacité d'un système à s'adapter à un changement d'ordre de grandeur de la montée en charge

minaux, ce qui représente un travail important d'administration et de certification, en plus de nécessiter des études spécifiques pour optimiser les performances de ce réseau avionique. L'ouverture de ce réseau à des données extérieures démultiplie le nombre de nœuds potentiels du réseau. A un niveau protocolaire supérieur à celui de l'AFDX, le protocole IP (niveau 3 "réseau" du modèle OSI) offre la possibilité d'augmenter la dimension des réseaux logiques en maintenant aux couches inférieures des réseaux physiques dimensionnés à des échelles acceptables. Le protocole IP est lié à des fonctionnalités de routage : alors qu'un "routage physique" (la commutation de trames) est effectué au niveau 2 par des commutateurs AFDX, un routage "logique" (le routage de paquets) est envisageable au niveau 3 à l'aide d'un composant réseau appelé "routeur IP". Nous travaillons à la conception, à la réalisation et à la validation d'un routeur de ce type.

2. Sûreté et sécurité aéronautique

Le développement des systèmes embarqués à bord des aéronefs est fortement contraint, que ce soit en terme de sûreté de fonctionnement qu'en terme de sécurité. En aéronautique, on parle de *safety* pour la sécurité ou encore la sûreté de fonctionnement des systèmes aéronautiques, c'est-à-dire les propriétés intrinsèques des systèmes leur permettant de "résister" aux dysfonctionnements. Le terme *security* désigne la sûreté des systèmes aéronautiques, c'est-à-dire les protections contre les menaces volontaires (attaques de pirates...). Nous emploierons les termes techniques anglais *safety* et *security* afin de ne pas les confondre avec les définitions françaises de *sécurité* et de *sûreté*.

Le domaine avionique est depuis toujours fortement contraint en terme de *safety*. Un dysfonctionnement général de l'appareil peut en effet conduire à sa destruction partielle ou totale, des vies humaines sont alors menacées. Différents standards aéronautiques spécifient les contraintes de *safety* devant être respectées afin que l'appareil soit autorisé à voler. Les logiciels avioniques embarqués critiques, tels que les pilotes automatiques par exemple, sont ainsi soumis au standard RTCA [2] DO-178B[13].

Le DO-178B définit 5 niveaux d'assurance concernant le développement des logiciels avioniques, appelés Design Assurance Level ou DAL. Le DAL-A est le niveau le plus exigeant en terme de contraintes et est utilisé pour les applications critiques dont les dysfonctionnements pourraient être catastrophiques, tandis que le DAL-E est un niveau non contraignant réservé aux applications ne pouvant pas avoir d'incidence sur la *safety* de l'appareil. Pour être embarqué et utilisé dans l'appareil, un logiciel doit être auparavant évalué suivant des critères de *safety* ; cette vérification appelée **"certification"** est effectuée par des organismes nationaux indépendants des développeurs. En France, la DGAC [3] est l'organisme de certification pour les systèmes aéronautiques.

A l'origine, seule la *safety* était prise en compte de manière approfondie par les avionneurs. Ceux-ci considéraient que la complexité des systèmes embarqués et la fermeture des réseaux avioniques garantissaient leur *security*. Depuis les attentats du 11 septembre 2011, la *security* est désormais devenue une préoccupation majeure pour la conception des systèmes et introduit de nouvelles contraintes et de nouvelles pratiques dans le monde aéronautique. Les produits sensibles doivent être validés d'un point de vue sécuritaire, cette validation de la *security* est appelée **"évaluation"**. Cependant, les processus de certification (pour garantir la *safety* du système) et d'évaluation (garantissant sa *security*) sont actuellement traités de manière parallèle et indépendente, certaines vérifications similaires étant ainsi effectuées deux fois. Ainsi, en plus de travailler sur les problématiques de la certification et de l'évaluation de notre routeur SNG, nous étudions celle de la mutualisation de ces deux processus.

3. Qualification d'outils pour la certification

La phase de certification d'un logiciel avionique est d'autant plus coûteuse, en terme de temps, de personnel et d'argent, que le niveau d'assurance exigé est élevé. Un pilote automatique d'avion qui doit être certifié au niveau DAL-A peut prendre plusieurs années à des centaines d'ingénieurs et coûter des millions de dollars ; la certification au niveau DAL-A nécessite en effet d'accomplir 66 tâches indépendantes de contrôle. Au contraire, le logiciel contrôlant l'affichage des signalisations "No Smocking" et "Fasten Seat Belt" ne nécessite que le DAL-D et sa certification n'est donc soumise qu'à 15 tâches de contrôle. Sa faible incidence sur la *safety* de l'appareil le rend ainsi beaucoup moins coûteux à développer.

Durant la conception, une technique pour réduire les coûts de certification consiste à automatiser certaines actions en utilisant des outils spécifiques garantissant certaines propriétés de *safety*. Par exemple, employer un compilateur qualifié DAL-A pour la conformité permet de garantir que le code binaire est conforme au code source et évite ainsi la tâche de "vérification de conformité du code binaire avec le code source" exigée pour la certification DAL-A. Cependant, l'outil (ici un compilateur) doit lui-même être vérifié pour s'assurer qu'il n'introduit pas de dysfonctionnement : la vérification d'un outil logiciel est appelée **"qualification"** et s'effectue de manière similaire à la certification de logiciel aéro-

2. RTCA : Radio Technical Commission for Aeronautics
3. DGAC : Direction Générale de l'Aviation Civile

nautique.

4. Une méthodologie élaborée pour le développement du routeur

Dans le cadre du développement du logiciel de notre routeur SNG, certifier un routeur existant aux niveaux exigibles par les avionneurs s'avère être très coûteux. C'est pourquoi nous avons cherché des solutions afin de développer un routeur de nouvelle génération en optimisant les coûts. Cela nous a conduit à créer une méthodologie de développement de logiciel aéronautique et à l'appliquer au cas de notre routeur SNG. Les objectifs de notre méthodologie sont la minimisation des durées de modélisation, de développement et de validation du routeur ainsi que la minimisation des efforts à fournir pour la certification (*safety*) et l'évaluation (*security*) tout en maximisant les niveaux de *safety* et de *security* apportées par le routeur SNG. Ainsi, le cas du routeur permet d'expérimenter et d'affiner concrètement notre méthodologie tandis que celle-ci permet un développement et une validation rapide et efficace de notre routeur SNG. Notre méthodologie présentée dans la prochaine section a été exposée de manière approfondie dans une publication présentée à la 30th Digital Avionics System Conference[15].

B. LA MÉTHODOLOGIE DE DÉVELOPPEMENT

1. La méthodologie en sept étapes

On suppose pour l'appliquer que la phase de spécification logicielle a été effectuée antérieurement ; ainsi, avant d'appliquer la méthodologie, l'architecte logiciel dispose d'un ensemble d'exigences spécifiant l'ensemble des services que le logiciel doit fournir ainsi que l'ensemble des contraintes qu'il doit respecter. Dans le cas de notre routeur SNG, cet ensemble d'exigences a été élaboré en accord avec Thalès Avionics et comprent notamment des exigences concernant le routage des paquets, d'autres pour leur sécurisation, d'autres enfin sur les performances attendues du produit.

La méthodologie peut être résumée par l'application successive des sept étapes suivantes illustrées par la figure 1.

Dans un premier temps, l'architecte logiciel effectue une partition de l'ensemble des exigences : il scinde cet ensemble en différents sous-ensembles disjoints tels que l'union des sous-ensembles donne l'ensemble initial d'exigences. Chaque sous-ensemble est alors associé à ce que l'on appelle une "partition logicielle". C'est l'étape de partitionnement. Dans le cadre du routeur SNG, trois partitions logicielles ont été extraites :

- les exigences de routage et de filtrage (Pfr4),
- les exigences de sécurisation des données (Pse),
- les exigences d'interfaçage du logiciel avec le matériel (Piface).

Ensuite, le comportement de chaque "partition logicielle" est représenté à l'aide de modèles graphiques de systèmes dynamiques et de machines à états. Ces modèles permettent aux développeurs de tester et valider dès la conception le fonctionnement du logiciel et de ses sous-parties. C'est l'étape de modélisation. Nous avons modélisé le routeur SNG à l'aide des outils Simulink[3] et Stateflow[4], deux toolkits fonctionnant avec Matlab.

Dans un troisième temps, les modèles sont convertis en code source à l'aide d'un générateur automatique qualifié de code. C'est l'étape de transformation. Nous avons généré le code source en langage C du routeur SNG à l'aide de l'outil Gene-Auto[14], un transformateur de modèle en code source libre et Open-Source.

Afin de fonctionner au sein d'un système, le code généré doit être lié à un système d'exploitation. Pour cela, du code additionnel appelé "glue code" est généré. Il contient notamment les liaisons entre les entrées/sorties du modèle et celles de la partition logicielle du système d'exploitation, ainsi que les informations d'ordonnancement des différentes partitions et le point d'entrée du code généré à l'étape précédente. C'est l'étape de collage. Dans le domaine de l'embarqué, certains systèmes temps-réel certifiables et évaluables existent, tels que PikeOS[5], commercialisé par Sysgo, que nous utilisons pour faire fonctionner les blocs logiciels du routeur SNG.

La cinquième étape est l'étape de compilation : les codes sources générés par les étapes 3 (transformation) et 4 (collage) sont compilés et liés ensemble en un code binaire par partition. L'usage d'un compilateur qualifié dans cette étape permet de réduire le coût de la certification du produit logiciel final.

Dans un sixième temps, l'architecte logiciel décrit la structure globale du produit logiciel en y précisant pour chaque partition logicielle le nombre d'instances devant s'exécuter simultanément, leur configuration, leur ordonnancement... Il regroupe dans une "image finale" ces informations, le noyau du système d'exploitation et l'ensemble des codes binaires de partition (générés à l'étape précédente). C'est l'étape d'intégration. Cette étape est souvent simplifiée par les outils de développement fournis avec le système d'exploitation, tel que le plugin Eclipse CODEO fourni avec Sysgo PikeOS.

Enfin, les développeurs peuvent télécharger cette image finale sur le matériel embarqué ou encore l'exécuter à l'aide d'un émulateur afin de vérifier son fonctionnement et valider ses performances. C'est l'étape d'exécution. Nous avons ainsi validé le fonctionnement correct du routeur avec quatre domaines communiquant en IPv4 et IPv6 grâce à l'émulateur qemu-x86[2].

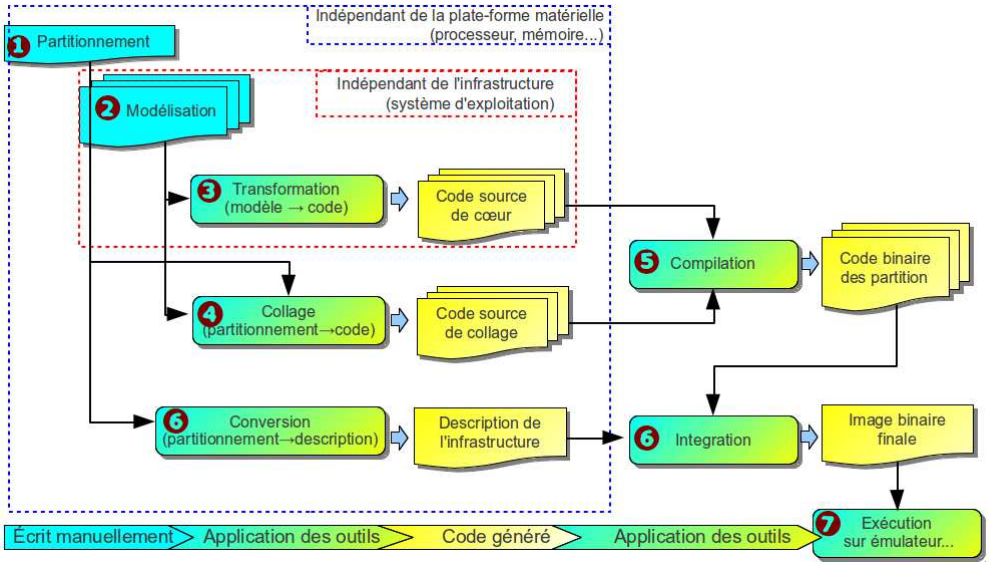

Figure 1 – Notre méthodologie appliquée à notre routeur SNG

2. Synthèse des bénéfices

Employer la méthodologie que nous avons créée présente de nombreux bénéfices. On peut citer notamment l'emploi de modèles de haut niveaux, réduisant la complexité de conception et de développement du logiciel et permettant de vérifier son fonctionnement au plus tôt. De plus, les modèles peuvent être transformés en codes sources en différents langages et facilement testés avec différents systèmes d'exploitation, sans devoir recommencer tout le codage. La décomposition fonctionnelle permet d'instrumenter [4] certaines parties du logiciel, d'étudier plus finement son fonctionnement et d'isoler plus rapidement les bugs qu'avec un code source monolythique. Enfin, les modèles peuvent être facilement réutilisés dans d'autres projets logiciels sans modification.

Nous avons appliqué la méthodologie au routeur SNG en nous appuyant sur une chaîne d'outils existants de modélisation (Simulink, Stateflow), de transformation (Gene-Auto) et un système d'exploitation (Sysgo PikeOS). Toutefois, notre méthodologie a été créée de manière à assurer une grande lattitude à l'équipe de développement concernant les outils employés. Nous envisageons ainsi de l'utiliser avec une autre chaîne d'outils tels que des outils de modélisation abstraite en langage B avec un compilateur CaML et un transformateur de modèle en code adapté, ou encore d'utiliser un système d'exploitation concurrent tel que le système temps réel VxWorks[6] de la société Windriver.

Les gains en terme de certification sont obtenus par l'emploi d'outils qualifiés DAL-A : 16 des 66 tâches exigées pour certifier au niveau DAL-A un logiciel sont réduites voire supprimées lorsque le transformateur et le compilateur sont qualifiés au même niveau. De même pour l'évaluation du produit final, l'emploi d'un système d'exploitation déjà évalué tel que Sysgo PikeOS facilite l'évaluation du produit logiciel final. Non seulement certaines parties de l'évaluation ne sont pas à ré-évaluer, mais en plus le contrôle par le système d'exploitation des communications entre partitions représente un avantage non négligeable pour garantir la *security* du logiciel.

C. LA SÉCURITÉ DU ROUTEUR

Après avoir parlé de la problématique de *safety* auquel est contraint notre routeur SNG et exposé la méthodologie de développement élaborée pour augmenter le niveau de confiance de notre routeur, nous allons maintenant évoquer les problématiques de *security* du routeur SNG. Les considérations de sécurité peuvent être catégorisées dans 2 classes : celles relatives aux flux transitant via le routeur et celles relatives au fonctionnement intrinsèque du routeur SNG.

1. Les Critères Communs

Différents standards industriels concernant la sécurité des logiciels existent. Les organismes législatifs internationaux aéronautiques n'en ayant pour l'instant privilégié aucun ni créé de nouveau dédié à l'avionique, nous avons choisi d'utiliser le standard ISO/CEI 15408 pour la *sécurity* du routeur SNG, celui-ci étant adapté à nos besoins et déjà utilisé en dehors du contexte aéronautique pour l'évaluation des systèmes d'information. Ce standard appelé Common Criteria for Information Technology Security Evaluation[1], ou plus souvent abrégé Common Criteria (CC), définit séparément :

– des exigences de sécurité que le produit doit

4. ajouter un observateur à un sous-système pour étudier ses interactions avec le reste du système

assurer (les exigences fonctionnelles ou *Security Functional Requirements*, SFR),
- des exigences de vérification que l'évaluateur doit valider concernant le produit (les exigences d'assurance ou *Security Assurance Requirement*, SAR).

Ces exigences de validation (SAR) sont souvent associées à des niveaux d'assurance, les Evaluation Assurance Level. Le premier niveau, et le moins contraignant, EAL1, exige de l'évaluation un rapport sur le produit contenant les spécifications techniques et la vérification d'éléments tels que la cohérence de la conception et la conformité des documents techniques. Les niveaux supérieurs sont de plus en plus contraignants, jusqu'au niveau maximal EAL7 qui nécessite une analyse approfondie du produit accompagnée, pour prouver les propriétés de *security*, de preuves formelles devant être automatiquement vérifiables.

2. Sécurité intrinsèque du routeur

Dans le cadre du routeur SNG, nous avons créé un Profil de Protection pour routeur sécurisé avec Thalès Avionics. Ce document, dont la forme est imposée par le standard, spécifie un ensemble d'exigences fonctionnelles de sécurité. Il est constitué de différentes sections. Dans une première partie introductive, les données relatives à l'identification du produit sont rappelées et complétées par les informations techniques relatives à la conception et à l'implémentation et utiles pour l'évaluation du logiciel de notre routeur.

Ensuite, le problème de sécurité propre au routeur SNG est exposé. Il y est notamment indiqué les parties sensibles du routeur (les données de configuration telles que les tables de routage statiques, les canaux de communication entre partitions...) avec le type de sensibilité (confidentialité des données, intégrité, authenticité...). Cette section est suivie d'une énumération des menaces pouvant altérer la sécurité du routeur, par exemple la corruption de la table de routage. Nous avons conduit pour cela une analyse des vulnérabilités du routeur SNG.

Dans un troisième temps, pour pallier ces menaces, nous avons énoncé les "objectifs de sécurité", spécifiant ce qui doit être fait pour rendre les menaces à l'encontre du routeur SNG inopérantes. Par exemple, il est imposé d'ajouter au routeur un mécanisme garantissant l'intégrité de sa table de routage. Enfin, les objectifs de sécurité sont complétés par des Security Functional Requirements (SFR). Ces SFR sont fortement contraints par le standard : le texte de l'exigence est imposé afin d'empêcher toute ambigüité de compréhension et de rendre toujours possible la vérification que le produit respecte l'exigence. Ces exigences de sécurité sont ensuite intégrées dans le processus de développement présenté précédemment et viennent compléter l'ensemble d'exigence utilisées durant l'application de notre méthodologie au routeur SNG.

Chaque section de notre Profil de Protection est accompagnée d'une sous-section exposant la traçabilité bidirectionnelle avec la section précédente. L'évaluateur peut ainsi vérifier que pour toute SFR, il existe au moins un objectif de sécurité associé et que tout objectif de sécurité est concrétisé par au moins une SFR.

3. Sécurisation des flux par le routeur

Par définition, un routeur relie entre eux plusieurs réseaux et permet donc la communication entre des nœuds de réseaux différents. Cette communication peut être sécurisée pour garantir, par exemple, la confidentialité (empêcher un troisième nœud de connaître le contenu du message échangé), l'intégrité (garantir l'absence d'altération du message) et l'authenticité (garantir la véracité de la source du message) des paquets échangés entre nœuds.

Nous avons spécifié et implémenté dans le routeur SNG des capacités de sécurisation pour interconnecter des réseaux de manière sécurisée : à l'image des Virtual Private Networks (VPN), il établit des canaux sécurisés de communication avec d'autres nœuds et les utilise pour transmettre les paquets ayant des besoins de sécurité. Nous avons choisi de nous baser pour cela sur un standard couramment employé pour la sécurité des réseaux au niveau 3 de la couche OSI : le framework IPsec (Internet Protocol Security[10]). Cette suite logicielle repose sur un ensemble de protocoles, pour établir dans un premier temps le canal sécurisé puis ensuite utiliser ce canal en garantissant les services de sécurité demandés (confidentialité, authenticité, intégrité...).

Le protocole Internet Key Exchange version 2 (IKEv2[8]) est un protocole couramment employé avec IPsec pour établir les canaux sécurisés : basé sur un échange de Diffie-Hellman de clefs, il permet la négociation des paramètres de sécurité d'une manière sûre entre les hôtes. Il fonctionne en coordination avec le protocole Internet Security Association Key Management Protocol (ISAKMP[12]) de la suite IPsec, protocole qui stocke sur chaque nœud la liste des canaux sécurisés.

Le canal précédemment établi à l'aide du protocole IKEv2 peut être utilisé pour échanger de manière sécurisée des paquets IP. Ces paquets sont alors encapsulés dans d'autres paquets IP (mode tunnel d'IPsec) ou leur entête est complétée avec des informations relatives à la sécurisation (mode transport d'IPsec). L'encapsulation est effectuée en ajoutant les informations d'autres protocoles qui effectuent concrètement la sécurisation. L'Encapsulating Security Payload (ESP[9]) est l'un de ces protocoles : il ajoute un identifiant de canal sécurisé, chiffre les données échangées pour assurer la confidentialité et ajoute des informations permettant de

garantir l'authenticité et l'intégrité des paquets IP.

Dans le cadre du routeur SNG, nous avons choisi d'utiliser les protocoles IKEv2 et ESP afin de sécuriser les données critiques. Le routeur SNG contient une partition logicielle nommée Pse qui est dédiée à la sécurisation. Le protocole IKEv2 est implémenté dans le routeur SNG à l'aide de machines à états transformées automatiquement en code source par Gene-Auto. La spécification du protocole ESP est concrétisée à la fois par une partie modélisée graphiquement pour le coeur du protocole et par du code logiciel écrit directement en langages de bas niveau pour les algorithmes de sécurité appliquées sur les données encapsulées.

D. CONCLUSIONS ET PERSPECTIVES

Dans cet article, nous avons présenté les besoins d'une nouvelle génération de routeur sûr et sécurisé destiné à l'aéronautique embarqué. Afin de concevoir ce produit logiciel, nous avons élaboré des spécifications logicielles pour le routeur SNG puis créé une méthodologie appropriée que nous avons mise en oeuvre jusqu'à la génération du logiciel et sa validation sur un émulateur de système embarqué. Notre méthodologie permet de simplifier le développement de logiciel aéronautique en maintenant un niveau élevé d'exigences en termes de certification et d'évaluation. Cette article a développé dans un second temps les problématiques plus spécifiques à la sécurité de notre routeur SNG et des flux qui les traversent ainsi que les solutions que nous y avons apportées.

Différentes pistes de travaux s'offrent à nous pour étoffer les fonctionnalités de notre routeur SNG. D'un point de vue sécurité, seuls trois algorithmes sont actuellements implémentés et devraient être complétés d'autres algorithmes prochainement. De plus, nous avons commencé à évaluer la compatibilité du fonctionnement de notre routeur SNG avec des routeurs commerciaux et des passerelles de sécurité VPN. La sécurisation actuelle est basée sur des canaux sécurisés de communication à deux extrémités (unicast), nous étudions la possibilité d'utiliser des canaux multicast où un nombre plus important de nœuds réseaux communiqueraient ensemble de manière sécurisée.

Concernant l'aspect génie logiciel, nous envisageons à court terme d'étudier les optimisations à apporter aux processus et aux outils. Ainsi, des tests préliminaires avec certains paramètres de compilation ont montré que des gains importants étaient possibles. Une autre piste consiste à employer d'autres langages pour le code source, notamment en testant prochainement le langage Ada en remplacement du C. Enfin, nous étudions d'autres chaînes d'outils afin d'apporter des gains en terme de vérification logicielle avec des méthodes formelles.

RÉFÉRENCES

[1] Common criteria for information technology security evaluation part 1 : Introduction and general model. Technical report, July 2009.

[2] QEMU, a generic and open source machine emulator and virtualizer, December 2011. http ://wiki.qemu.org/.

[3] Simulink Dynamic system modeler toolkit for Matlab, December 2011. http ://www.mathworks.com/products/simulink/.

[4] Stateflow State Machine modeler toolkit for Simulink, December 2011. http ://www.mathworks.com/products/stateflow/.

[5] Sysgo PikeOS Real-Time Operating System, December 2011. http ://www.sysgo.com/products/pikeos-rtos-and-virtualization-concept/.

[6] WindRiver VxWorks Real-Time Operating System, December 2011. http ://www.windriver.com/ products/vxworks/.

[7] R. Garside and F. Pighetti. Integrating modular avionics : A new role emerges. *Aerospace and Electronic Systems Magazine, IEEE*, 24(3) :31 –34, march 2009.

[8] C. Kaufman and P. Hoffman. Internet Key Exchange (IKEv2) Protocol. RFC 5996 (Proposed Standard), September 2010. http ://www.ietf.org/rfc/rfc5996.txt.

[9] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), December 2005. http ://www.ietf.org/rfc/rfc4303.txt.

[10] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005. http ://www.ietf.org/rfc/rfc4301.txt.

[11] Ian Land and Jeff Elliott. *Architecting ARINC 664, Part 7 (AFDX) Solutions*. Xilinx, May 2009.

[12] D. Piper. The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407 (Proposed Standard), November 1998. Obsoleted by RFC 4306, http ://www.ietf.org/rfc/rfc2407.txt.

[13] RTCA. Software considerations in airborne systems and equipment certifications, DOD-178B. Technical report, RTCA, Washington D.C., 1992.

[14] Andreas Toom and al. Gene-auto : An automatic code generator for a safe subset of simulink/stateflow and scicos. In *4th European Congress ERTS Embedded Real Time Software*, 2008.

[15] Antoine Varet and Nicolas Larrieu. New methodology to develop certified safe and secure aeronautical software an embedded router case study. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 7C6–1 –7C6–15, oct. 2011.