A Formal Security Model for Verification of Automotive Embedded Applications

Gabriel Pedroza, Ludovic Apvrille, Renaud Pacalet
Institut Telecom, Telecom ParisTech, LTCI CNRS
2229, Routes des Crêtes, B.P. 193 F-06904 Sophia Antipolis, France
Email: {gabriel.pedroza, ludovic.apvrille, renaud.pacalet}@telecom-paristech.fr

***Abstract*— Intelligent Transport Systems (ITS) arose several years ago as a promising solution to decrease road casualties [1]. ITS are based upon heterogeneous wireless networks of vehicles and road side infrastructures. In such critical vehicular embedded systems, the safety is at stake, and so, the security of car-to-car and car-to-infrastructure communications shall be carefully taken into account, first for safety issues, but also for privacy issues. We propose a validation methodology that assists the design of such embedded systems, and based on a Formal Security Model that mainly targets model checking. If modeling of embedded in-car applications is a major issue due to system constraints and complexity, formal validation of such models shall offer a higher level of guarantee. Apart from its formal semantics, another strength of our approach relies on the decoupling between system design and security issues (e.g., attacks, requirements, and so on) whilst all are integrated in the same framework.**

I. Introduction

Intelligent Transport Systems (ITS) intend to dramatically decrease road casualties. One of the main ideas is to have cars and road infrastructures communicate and collaborate so as to detect and avoid dangerous situations. Those communications rely on wireless networks. Because those networks and systems could be targeted by attackers, the security of vehicular embedded systems is an ongoing trend.

The development of protected in-car applications is a challenging task. Indeed, to support current vehicle applications, cars are commonly equipped with up to 70 controllers called Electronic Control Units (ECU) [2]. ECUs are interconnected through several buses (CAN, FlexRay) thus forming highly complex networks. The actual protection for in-car safe systems like centralized car closure or mileage counter protection mainly relies on software-based solutions that are installed in the core CPU. Additionally, gaining access to the core CPU makes it possible to manipulate and compromise car operations [2]. In this context, the successful implementation of ITS relies on protection of in-car applications. If the design of embedded systems has been studied for years, the development of secured embedded system, taking into account both hardware and software levels, and integrating all methodological stages is still an open issue, at least in the vehicular domain. For example, solutions proposed in [3] to [12] lack either flexibility or formality, therefore leading to strong limitations on proposed design solutions.

The purpose of this paper is a methodology which aims to provide a modeling and verification method integrated with usual embedded systems design methodologies. To achieve this, we introduce a Formal Security Model (FSecM). FSecM borrows elements from other security frameworks and approaches, for instance attacks and security requirements modeling. FSecM also introduces new relevant elements: it mainly includes a first attempt of formal security language unification. Our security model also addresses dependencies between security properties.

The FSecM is part of a global methodology which includes in-car system representation and formal verification based on model checking. This methodology is briefly presented in section III. A complementary but relevant part of our approach is the instantiation of the proposed methodology: one possible instantiation is presented in section IV. We finally conclude this paper (section V).

II. Formal Security Model (FSecM)

The FSecM integrates four entities: *System Attacks, Security Requirements, Formal-based Security Properties and Security Dependencies* (See Figure 1). In next subsections a brief description of these security entities is presented.

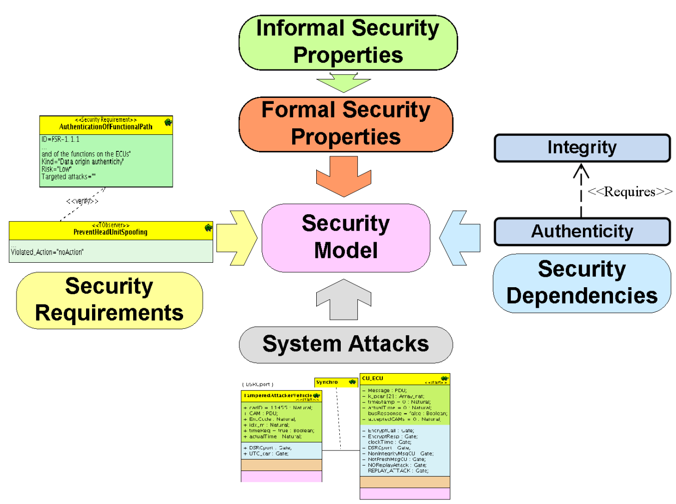

Fig. 1. Security Model

A. System Attacks

FSecM includes System Attacks. The rationale for Attacks comprises Dolev-Yao threat model instantiation [13]. This approach supposes that an attacker can listen, overhear, intercept and alter agent's exchanges and its power is only limited by restrictions imposed by cryptographic protocols. Attacks are seen as any possible violation of correct system operation. Identified attacks are usually modeled through malicious agents that interact with the system model, and that are expressed in the same language [14]. The modeling of known attacks usually gives a first insight for identifying unknown attacks. The construction of a generic attack model is a paramount task that should be ideally included. However it is out of the scope of this paper.

B. Security Requirements

Security Requirements (SR's) establish protection requirements for in-car HW-SW assets and components. SR's can be obtained from translation of a technical/security specification (which is written in plain language) or as protection against identified attacks. Attacks can be derived from risk and threat analysis as well as directly from Use Case scenarios. SR's can be categorized in hierarchy graphs whose nodes are informal security requirements. Indeed, the graph determines relationships between SR's whose semantics needs to be formalized.

C. Formal Security Properties

Formal Security Properties (FSP's) are the core of our FSecM. We claim that formal-based definitions can provide a unified-consistent and efficient security framework. FSP's are defined from a semantic analysis that includes categorization, abstraction and unification of informal security properties. From this analysis, informal definitions for *Integrity, Authenticity, Freshness, Non-Repudiation, Controlled Access, Availability, Confidentiality, Anonymity and Privacy* were derived. Several temporal logic languages still need to be evaluated to determine the best candidate for representing them at formal verification stage [15], [16], [19]. Since we target model checking we currently explore expressiveness capabilities of CTL.

D. Security Dependencies

Security dependencies are meant to offer a support for formal verification purposes. The security dependencies are derived from a logical analysis of security definitions. This analysis takes into account the verification context:

1) *Verifier Viewpoint:* A viewpoint is related to the verifier's knowledge. The property may be verified with respect to the viewpoint of different system Verifiers. Local as well as global viewpoints may modify the property representation. We exemplify this by considering an external message that arrives to a target vehicle. To verify *Integrity* the Verifier may assume that the original message is known or contrarily that it is unknown. Hence, property representation may accordingly change thus reflecting different agent's knowledge.
2) *Target of the Property:* The modeling entity(ies) to which the property is meant to be verified. A property may be addressed for verifying different elements in the model. Property representation and its dependencies may evolve according to the different targets. Thus, for instance the verification of message *Integrity* relies on original and received messages whereas platform *Integrity* relies on non altering of code and registers.
3) *Assumptions:* Security properties may be implicitly/explicitly assumed in the system model. For instance when two communicating elements are encapsulated in the same SoC, *Integrity* of exchanges might be assumed by default. Related assumptions like channel integrity protection and message immunity against transmission delays may also modify security dependencies.

Since verification of properties may be influenced by the context, several schemes of security dependencies may arise; global and local schemes for formal verification might be necessary. We currently work on the establishment and formalization of these security dependencies.

E. FSecM integration

A SR specification conforms a hierarchy graph of informal security requirements (See section II-B). Each lowest dependency node of a SR can be related to a Security Property in the FSecM thus achieving formal based SR's. Afterwards, dependencies between security properties determine hierarchical order for verification; if *Authenticity* is going to be verified and *Authenticity* has a dependency with respect to *Integrity* then *Integrity* should be verified first. Modeling assumptions may provoke that certain security properties are held by default. In such case those properties are ensured and don't require additional verification. Last, once the system model satisfies a security property it must resist related attacks.

III. Global Validation Methodology

The global validation methodology is mainly intended to formally represent and validate security properties in automotive applications. The flow of this methodology is presented in Figure 2. The methodology comprises translation of the technical specification to the model (box 1), and the decoupling between security and other concerns (boxes 2 and 3). Indeed, since we target model checking, the system must be represented in a formal framework, for instance by Timed Automata [17] or Kripke structures [18]. We indeed consider that the decoupling of system and security models is an initial step in order to achieve model correctness. By this separation we aim to provide enough flexibility for SW-HW modeling and to prevent that system model is biased by security specification, and conversely. As a consequence of this decoupling, the process of integrating security model to system model should be as automated as possible (box 6). Model simplifications and refinements are suggested as a way to simplify verification process and to deal with state explosion problem (boxes 4 and 5), but preservation of relevant model properties is a paramount issue. Since the state explosion problem may limit verification capabilities even after model simplification, we propose a hybrid method for validation (box 7). This hybrid method may combine formal as well as heuristic techniques, attack tests and simulation. Security flaws are finally identified (box 8) thus starting a model improvement process until all security requirements are satisfied.

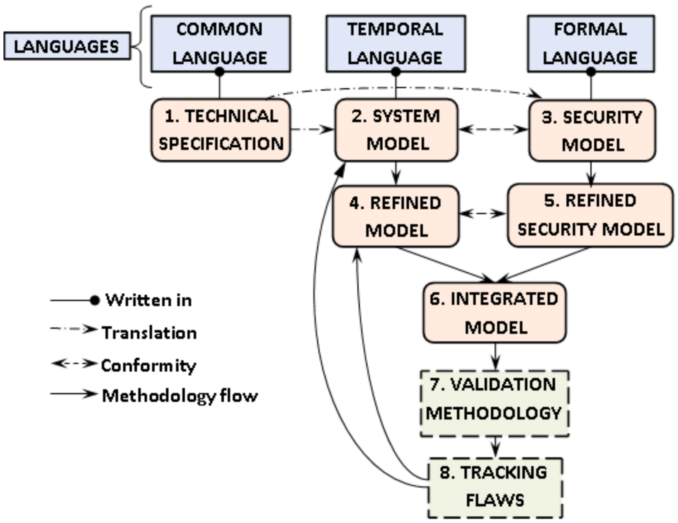

Fig. 2. Methodology Flow

IV. Example of FSecM

An example of FSecM integration has been made using TTool [20]. This tool targets system formal verification through the implementation of formally defined UML profiles, including the Timed UML Real-Time Language Environment (TURTLE) profile [21]. TTool relies on SysML Requirement Diagrams to represent Security Requirements. Since attacks are closely related to system operation, they are represented as classes modeled along with system classes, within TURTLE class diagrams. The latter define the system architecture based on TURTLE classes, and on communications between those classes through composition operators. Internal behavior of each TURTLE class must be expressed with a UML Activity Diagram. TTool automatically translates TURTLE class and Activity Diagrams to a specification in a formal language (LOTOS, RT-LOTOS, UPPAAL). TURTLE offers a good level of abstraction thus allowing a flexible HW-SW modeling. Formal Security Properties as well as Security Dependencies still need to be integrated in TTool as an extension of the TURTLE profile.

Our FSecM example relies on the work that has been developed in the scope of the EVITA project [22]. The target of verification is based upon a model of a Local Danger Warning (LDW) scenario [23]. In this scenario a vehicle aims to securely broadcast a Cooperative Awareness Message (CAM) among neighborhood vehicles. The goal is to prevent cars about of a casualty on the road. In this example the FSecM is composed by:

1) A tampered external vehicle which is represented in the model. This attacker vehicle broadcasts older-faked messages to the neighborhood vehicles.
2) A set of Security Requirements that should be accomplished to achieve trust exchanges. To prevent attacker's behavior we should address specific security requirements, for instance *Freshness Requirement for message exchanges*.
3) A set of formal security properties. In this case a formal definition for *Freshness* is required.
4) The verification dependencies for *Freshness*. The Verifier viewpoint considers that the original CAM message is unknown. The attacker model supposes possible message delays and altering. Additionally, since it is assumed that *Freshness* is verified based on message contents consequently message *Integrity* should be verified before *Freshness*.

V. Conclusions

Dealing with both security and safety constraints when designing vehicular systems is a challenging task that we propose to address with a new framework called FSecM. Through FSecM and the proposed validation methodology, we address issues of Intelligent Transport Systems validation: consistencies in security approaches, flexibility for HW-SW modeling, process automation and formal verification of complex embedded systems.

If we have already successfully experimented that framework over some uses cases of the EVITA project, methodological stages of the FSecM still need to be more closely defined.

Acknowledgment

This work has been performed in the scope of the EVITA project financed by the European Commission.

References

[1] *Objectives of the EVITA project*, in http://www.evita-project.org/objectives.html.

[2] T. Kosch, *Local Danger Warning based on Vehicle Ad-hoc Networks, Prototype and Simulation*, Proceedings of 1st International Workshop on Intelligent Transportation (WIT), Hamburg, Germany, 2004.

[3] M. J. Toussaint, *A New Method for Analyzing the Security of Cryptographic Protocols*, IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, June 1993.

[4] Denis Treck and Borka Jerman Blazic, *Formal Language for Security Services base Modelling and Analysis*, Elseiver Science Journal, Computer Communications Vol. 18, No. 12, 1995.

[5] Michael Drouineaud, Maksym Bortin, Paolo Torrini, Karsten Sohr, *A First Step towards Formal Verification of Security Policy Properties for RBAC*, Proceedings of the Fourth International Conference on Quality Software (QSIC'04), IEEE 0-7695-2207-6/04, 2004.

[6] Clare Dixon, Mari-Carmen Fernández Gago, Michael Fisher and Wiebe van der Hoek, *Using Temporal Logics of Knowledge in the Formal Verification of Security Protocols*, Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME'04), IEEE 1530-1311/04, 2004.

[7] Achim D. Druker and Burkhart Wolff, *A Verification Approach to Applied System Security*, Internation Journal in Software Tools and Technology, Vol. 7, pp. 233-247, Springer-Verlag, 2005.

[8] Alessandro Aldini, Marco Bernardo, *A Formal Approach to the Integrated Analysis of Security and QoS*, Reliability Engineering and System Safety, Vol. 92,pp. 1503-1520, Elseiver, 2007.

[9] Genge Béla and Haller Piroska, *A Modeling Framework for Generating Security Protocol Specifications*, 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE 978-0-7695-3523-4/08, 2008.

[10] Antonio Maña and Gimena Pujol, *Towards Formal Specification of Abstract Security Properties*, The Third International Conference on Availability, Reliability and Security, IEEE 0-7695-3102-4/08, 2008.

[11] Riham Hassan, Shawn Bohner, Sherif El-Kassas and Michael Hinchey, Proceedings of the 42nd Hawaii International Conference on System Sciences, IEEE 978-0-7695-3450-3/09, 2009.

[12] Yomna Ali, Sherif El-Kassas, Mohy Mahmoud, *A Rigorous Methodology for Security Architecture Modeling and Verification*, Proceedings of the 42nd Hawaii International Conference on System Sciences, IEEE 978-0-7695-3450-3/09, 2009.

[13] D. Dolev and A. Yao. *On the security of public key protocols*, in Proceedings of the IEEE Transactions on Information Theory, 29(2), 1983.

[14] A. Maturia, A.R. Singh, P.V. Sharavan, R. Kirtankar, *Some New Multi-Protocol Attacks*, in proceedings 15th International Conference on Advanced Computing and Communications, 0-7695-3059-1/07, IEEE, 2007.

[15] Z. Huzar, J. Maggot, *Real-Time and Performance Evaluation Extensions of Specification Language LOTOS*, 0-8186-6902-0/95, IEEE, 1995.

[16] J-P. Courtiat, C.A.S. Santos, C. Lohr, B. Outtaj, *Experience with RT-LOTOS, a Temporal Extension of the LOTOS Formal Description Technique*, in Computer Communications Vol. 23, Elsevier Science, pp. 1104-1123, 2000.

[17] A. Furfaro and L. Nigro, *Temporal Verification of Communicating Real-Time State Machines Using Uppaal*, IEEE International Conference on Industrial Technology, Rende, Italy, Dec 2003.

[18] E.M. Clarke Jr., O. Grumberg and D.A. Peled, *Model Checking*, Ed. The MIT Press, Cambridge Massachusetts, London, England, 1999.

[19] G. Behrmann, A. David and K.G. Larsen, *A Tutorial on Uppaal*, in http://www.uppaal.com/.

[20] *TTool: The TURTLE Toolkit*. In http://labsoc.comelec.enst.fr/turtle/ttool.html.

[21] L. Apvrille, J.-P. Courtiat, C. Lohr, and P. De Saqui-Sannes, *TURTLE: A Real- Time UML Profile Supported by a Formal Validation Toolkit*, In IEEE transactions on Software Engineering, volume 30, pages 473-487, Jul 2004.

[22] The website of the EVITA project in http://www.evita-project.org/.

[23] E. Kelling, M. Friedewald, T. Leimbach, M. Menzel, P. Säger, H. Seudie, and B. Weyl, *Specification and Evaluation of e-Security Relevant Use Cases*, Technical Report Deliverable, D2.1, EVITA Project, 2009.