Model-Driven Engineering for Designing Safe and Secure Embedded Systems

Ludovic APVRILLE, Letitia W. LI
Institut Mines-Telecom
Telecom ParisTech, CNRS/LTCI
Sophia Antipolis, France
Email: {ludovic.apvrille, letitia.li}@telecom-paristech.fr

Yves ROUDIER
EURECOM
Sophia Antipolis, France
Email: Yves.Roudier@eurecom.fr

***Abstract*—The communication capabilities of recent embedded systems offer more opportunities for attack to cyber criminals. Moreover, those attacks may compromise the safety of these systems. SysML-Sec is a SysML-based environment for the design of such embedded systems with safety and security features.**

The paper focuses on the SysML-Sec methodology containing the following stages: assumptions, requirements, attacks, partitioning, software design and software deployment. Our method is supported by TTool, and offers a press-button approach for formal proof of safety and security. Previous projects and case studies modeled and validated with SysML-Sec range from automotive systems, drone systems, information systems (e.g., the analysis of malware targeting banking systems), industrial systems (Analysis of SCADA malware), and more generally, security protocols.

I. Introduction

Inter-connected embedded systems and cyber-physical systems offer more opportunities for attack to cyber criminals. To cite only a few examples of recent attacks on such connected systems, we can mention ADSL routers [1], mobile&smart phones [2], avionics [3], automotive systems [4], medical appliances such as the Hospira Symbiq drug pump [5], and smart objects e.g. the recent vulnerability disclosed on the Fitbit [6]. Such attacks also target industrial systems whose sensors are increasingly commonly connected with vulnerable information systems, as demonstrated by Stuxnet, Flame, and Duqu [7]. Attacks threaten the dependability of such systems with various objectives ranging from extortion to terrorist acts.

System complexity, notably in terms of code size, distribution, and heterogeneity, is a major risk factor to safety and security. This issue can only be mitigated by the verification of all the interactions between functional and non-functional requirements throughout a system's development cycle. The SysML-Sec environment aims to improve the design of such complex systems with respect to their performance, safety, and security [8]. SysML-Sec addresses system development starting from requirements and possible attacks, continuing into software/hardware partitioning, and further progressing into the design of software components. SysML-Sec is implemented within a free/open-source tookit named "TTool" [9]. TTool offers diagramming capabilities as well as safety and security proofs at the push of a button. This paper summarizes the SysML-Sec methodology, its model-based approach, and its support by TTool.

II. SysML-Sec Methodology

A. Methodological stages

The methodology of SysML-Sec, summarized in Figure 1, addresses all stages that are commonly used to design an embedded system.

The requirements/attacks stage intends to identify and analyze both requirements and attacks together with the main application functions (Functional View). Formally defined attack graphs [10] are used to capture attack scenarios, which commonly rely on the exploitation of a combination of vulnerabilities. Once defined, these graphs can be easily migrated for reuse in analysis of other systems. Attacks can be linked together in order to assess the impact of a specific vulnerability and the need to address it at the risk assessment phase, e.g., once a mapping is under evaluation.

The SW/HW Partitioning phase follows the Y-Chart approach with the modeling of logical tasks/channels (Functional view), candidate architectures (Architectural view), and mapping (Mapping view) of tasks and channels onto the architecture. Candidate architectures consist of CPUs, hardware accelerators, buses, bridges, and memory elements. Tasks abstract the system behavior, and define concepts of execution time and inter-task communications. The attributes passed between tasks are abstracted to consider only the size of the data transferred or stored. Once system partitioning has been completed, software

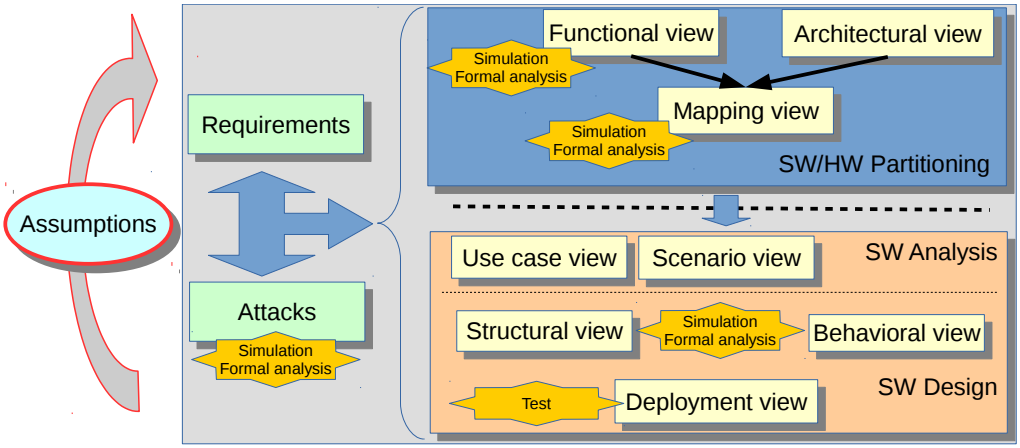

Fig. 1. System design with the SysML-Sec Methodology

design can begin. The goal of the software analysis and design stages is to develop the software components implementing the functions mapped onto processors at the previous stage. Functions to be implemented are first analyzed with SysML-based use case and scenario views to determine a software design in terms of safety or security-related SysML blocks (Structural view) and their interactions, e.g., security protocols. During the design, software components/blocks are progressively refined until the point where executable code generation is feasible. This refinement also includes adding security-related functions (Behavioral view), e.g., cryptographic algorithms, key management policies, and filtering policies. The deployment view specifies which executable code can be generated: local execution, specific board (hardware architecture previously defined at mapping stage), or virtual prototyping environment [11] (abstracting the same hardware architecture).

Iterations over the complete method are assumption-driven [12]. More precisely, the system specification is first limited in scope, then progressively enlarged to include further details. Assumptions expressing the scope of the specification are explicitly described in a Modeling Assumption Diagram. The safety and security assumptions are included within this model, thus impacting the safety and security proofs at each new iteration. Advancing from version n to version $n+1$, modifications on diagrams and properties are thus traced within the Modeling Assumption Diagrams.

B. *Verification of properties along the SysML-Sec method*

During the functional stage, simulations and (formal) verifications are performed in order to identify safety-related issues, e.g., deadlock situations, non-reachability of error states, etc. Functional models are untimed, which means that no performance study can be conducted on them. However, the mapping of functions over execution nodes gives to the former a logical and physical execution time. Thus, post-mapping simulations and formal verifications are intended to demonstrate the system performance on the selected hardware architecture, including the study of latencies, the load of processor and buses, and communication time. The results are determined how the logical functions - including the security mechanisms - are linked to the underlying hardware. For example, a given security protocol may impact a bus load, a cryptographic function may impact a processor load, and both can consequently increase the overall system latency. The mapping scheme, for example, mapping a cryptographic function over a hardware accelerator, or instead on a general-purpose processor, also impacts system latencies. The performance analysis thus intends to study both safety and security functions mapped on different conditions [13]. The result of this study is a hardware/software architecture that complies with both safety and security requirements, and that can resist attacks of the given risk level.

During the first iterations of the software design, simulation can be used to debug the models [14]. When the model has been refined to include all relevant behavior, formal verification can also be used to assess safety properties [15] [16] (e.g., the reachability of a given state, or its liveness). Security proofs can also be performed on software design diagrams. The latter are transformed into a security-oriented formal specification in pi-calculus [17]. This specification is automatically sent by TTool to ProVerif, along with the confidentiality and authenticity properties to be proved [18]. These properties may refer

to the confidentiality of a SysML block attribute or the authenticity of a message exchanged by two blocks.

When the model is too large to be verified, model-to-code transformations are used to perform security and safety tests on the code itself.

III. TTool

The free and open-source TTool software supports all SysML-Sec methodological stages, including model capture, simulation, and verification. In fact, TTool is a multi-profile toolkit whose main strength is to offer a press-button approach for performing simulation and formal proofs from models. TTool supports the automatic proof of both safety and security properties in LOTOS [19], UPPAAL [16] and ProVerif [18]. For user convenience, results of the verification are back-traced to the graphical models. For example, Figure 2 displays a subset of the software design of HTTPS/TLS. The confidentiality and authenticity proofs have been conducted on this SysML-Sec model, and the results have been back-traced in the form of a lock added adjacent to the relevant model element, such as an authenticity pragma or block attribute whose confidentiality property has been queried. A green lock indicates the property is satisfied, a grey lock indicates the property cannot be proved, and a red lock indicates the property is not satisfied.

IV. Related work

From our experience, partitioning is a central issue in embedded systems. Achieving a correct partitioning that adheres to safety requirements necessitates that the impact of security mechanisms be understood and quantified as early as possible. SecureUML, for example, extends UML to provide capabilities to consider security in a model and verify its security requirements. [20]. We note that only a few authors, notably Eames and Moffet [21], and more recently Pietre-Cambacédès [22] and Raspotnig [23], have dealt with the relationships between security and safety requirements. For instance, the last two authors discuss quite thoroughly the relationships that can be established between security and safety requirements. In particular, these studies can describe conflicts, but also reinforcement relationships (when safety and security concur towards the same design), or conditional dependence. We think that obtaining similar descriptions within SysML-Sec would require the engineering methodology to be extended with an additional feedback interaction from all engineering phases to the specification phase: for instance, the satisfaction of safety requirements should be checked based on the security mechanisms introduced before any further safety mechanism would be introduced. We did not evaluate any such methodological step. However, we plan to investigate these issues in the future.

V. Conclusion - Future Work

Attacks are now widely conducted on embedded and cyber-physical systems. Unfortunately, current software engineering methods first focus on the safety and a fast time-to-market of these systems, with considerations for security as an after-thought.

SysML-Sec addresses this issue with a model-driven engineering approach combining semi-formal specifications of both safety and security features and properties at any given development cycle phase. SysML-Sec is supported by a free/open-source toolkit, TTool. Simulations and formal proofs on models can be easily conducted with TTool, in order to assess function, architectural choices and software design choices, in terms of performance, safety properties, and security properties.

SysML-Sec has been previously used in the design and evaluation of embedded systems (e.g., automotive architectures), security protocols (TLS), analysis of complex attacks (Stuxnet, Zeus/Zitmo).

Our future work concerns the enhancement of security modeling and validation capabilities at the partitioning stage. The partitioning of tasks onto execution nodes impacts security, as data sent between two tasks mapped to the same CPU cannot be intercepted. Such consideration of security may reduce latencies or hardware in removing the need to encrypt certain data. Furthermore, channels can be mapped onto either public or private buses. Similarly to the ProVerif-based security proof of software design, we indeed intend to define and implement a mapping-to-proverif model transformation to consider security. We also intend to introduce a security-oriented modeling assistant to help identify threats, their countermeasures, and supporting hardware and software architectures. This assistance might be based on a library of modeling patterns.

References

[1] F. Assolini, "The Tale of One Thousand and One DSL Modems, kaspersky lab," Oct. 2012. [Online]. Available: http://www.securelist.com/en/blog/208193852/The_tale_of_one_thousand_and_one_DSL_modems

[2] D. Maslennikov, "Russian cybercriminals on the move: profiting from mobile malware," in *The 20th Virus Bulletin Internation Conference*, Vancouver, Canada, Oct. 2010, pp. 84–89.

[3] A. Costin and A. Francillon, "Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices," in *BLACKHAT 2012, July 21-26, 2012, Las Vegas, NV, USA*, Las Vegas, USA, 07 2012.

[4] T. Hoppe, S. Kiltz, and J. Dittmann, "Security Threats to Automotive CAN Networks - Practical Examples and Selected Short-Term Countermeasures," *Rel. Eng. & Sys. Safety*, vol. 96, no. 1, pp. 11–25, 2011.

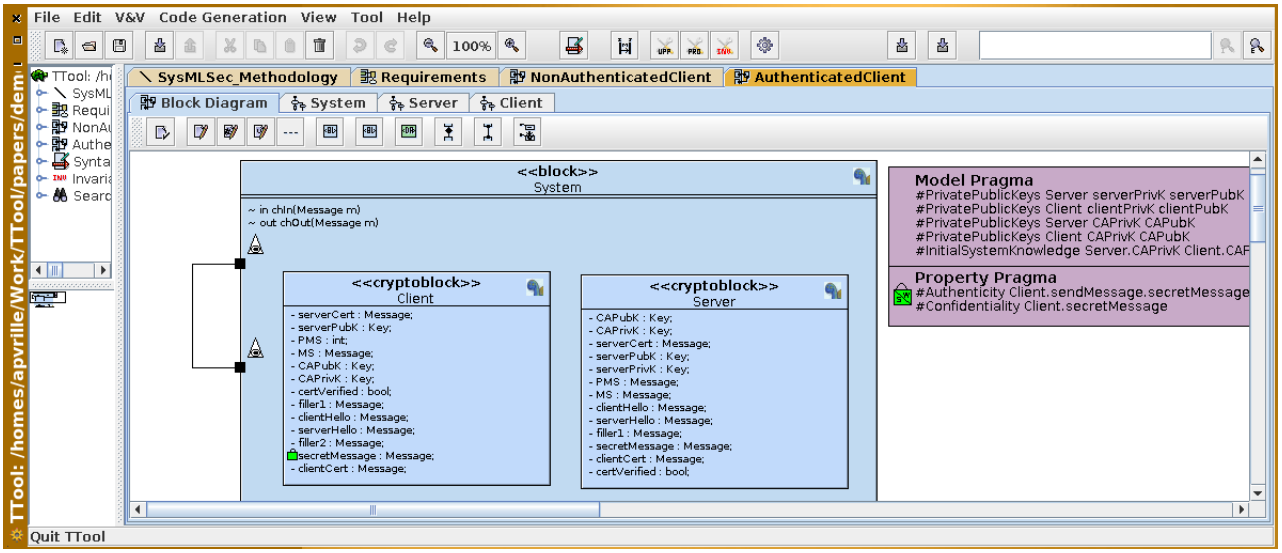

Fig. 2. SysML-Sec model in TTool featuring the results of a security formal verification

[5] ICS-CERT, "Hospira lifecare pca infusion system vulnerabilities, advisory (icsa-15-125-01b)," https://ics-cert.us-cert.gov/advisories/ICSA-15-125-01B, Jun. 2015. [Online]. Available: https://ics-cert.us-cert.gov/advisories/ICSA-15-125-01B

[6] A. Apvrille, "Geek usages for your fitbit flex tracker hack.lu, luxemburg, october 2015," Slides at fr-amadrive.org/index.php/s/Wk6nxAKMpVTdQl4, Oct. 2015.

[7] D. Maynor, "Scada security and terrorism: We're not crying wolf!" in *Invited presentation at BlackHat BH 2006. Presentation available at: https://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Maynor-Graham-up.pdf*, USA, 2006.

[8] L. Apvrille and Y. Roudier, *Model-Driven Engineering and Software Development*. Switzerland: Springer International Publishing, 2016, ch. Designing Safe and Secure Embedded and Cyber-Physical Systems with SysML-Sec, pp. 293–308.

[9] L. Apvrille, "TTool," ttool.telecom-paristech.fr, Dec. 2003. [Online]. Available: ttool.telecom-paristech.fr

[10] L. Apvrille and Y. Roudier, "Sysml-sec attack graphs: Compact representations for complex attacks," in *The Second International Workshop on Graphical Models for Security (GraMSec 2015)*, vol. 9390. Verona, Italy: Springer, LNCS, Jul. 2015, pp. 35–49.

[11] D. Genius and L. Apvrille, "Virtual yet precise prototyping: An automotive case study," in *8th European Congress on Embedded Real Time Software and Systems (ERTS2'2016)*, Toulouse, Jan. 2016.

[12] P. De Saqui-Sannes and L. Apvrille, "Making modeling assumptions an explicit part of real-time systems models," in *8th European Congress on Embedded Real Time Software and Systems (ERTS2'2016)*, Toulouse, France, Jan. 2016.

[13] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "C2x communication: Securing the last meter," in *The 4th IEEE International Symposium on Wireless Vehicular Communications: WIVEC2011*, San Francisco, USA, Sep. 2011.

[14] L. Apvrille and P. De Saqui Sannes, "AVATAR/TTool : un environnement en mode libre pour SysML temps réel," *Génie Logiciel*, no. 98, pp. 22–26, Sep. 2011.

[15] L. Apvrille and P. De Saqui-Sannes, "Requirements analysis," *Embedded Systems: Analysis and Modeling with SysML, UML and AADL*, 2013.

[16] J. Bengtsson and W. Yi., "Timed automata: Semantics, algorithms and tools," in *Lecture Notes on Concurrency and Petri Nets*. W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, 2004, pp. 87–124.

[17] F. Lugou, L. W. Li, L. Apvrille, and R. Ameur-Boulifa, "Sysml models and model transformation for security," in *Conferénce on Model-Driven Engineering and Software Development (Modelsward'2016)*, Rome, Italy, Feb. 2016.

[18] B. Blanchet, "Automatic Verification of Correspondences for Security Protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 363–434, Jul. 2009.

[19] ISO-LOTOS, "A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour," in *Draft International Standard 8807, International Organization for Standardization - Information Processing Systems - Open Systems Interconnection*, Geneva, July 1987.

[20] J. Jürjens, "Umlsec: Extending uml for secure systems development," in *Proceedings of the 5th International Conference on The Unified Modeling Language*, ser. UML '02. London, UK, UK: Springer-Verlag, 2002, pp. 412–425. [Online]. Available: http://dl.acm.org/citation.cfm?id=647246.719625

[21] D. P. Eames and J. D. Moffett, "The integration of safety and security requirements," in *SAFECOMP*, 1999, pp. 468–480.

[22] L. Pietre-Cambacedes and M. Bouissou, "Cross-fertilization between safety and security engineering," *Rel. Eng. & Sys. Safety*, vol. 110, pp. 110–126, 2013.

[23] C. Raspotnig and A. L. Opdahl, "Comparing risk identification techniques for safety and security requirements," *Journal of Systems and Software*, vol. 86, no. 4, pp. 1124–1151, 2013.