Security Modeling for Embedded System Design

Letitia W. Li[1,2], Florian Lugou[1], and Ludovic Apvrille[1]

[1] Télécom ParisTech, Université Paris-Saclay
450 route des Chappes, Sophia Antipolis, France,
{letitia.li,florian.lugou,ludovic.apvrille}@telecom-paristech.fr
[2] Institut VEDECOM,
77 rue des Chantiers, Versailles, France
letitia.li@vedecom.fr

Abstract. Among the many recent cyber attacks, the Mirai botnet DDOS attacks were carried out using infected IoTs. To prevent our connected devices from being thus compromised, their security vulnerabilities should be detected and mitigated early. This paper presents how the SysML-Sec Methodology has been enhanced for the evolving graphical modeling of security through the three stages of our embedded system design methodology: Analysis, HW/SW Partitioning, and Software Analysis. The security requirements and attack graphs generated during the Analysis phase determine the sensitive data and attacker model during the HW/SW Partitioning phase. We then accordingly generate a secured model with communication protection modeled using abstract security representations, which can then be translated into a Software/System Design Model. The Software Model is intended as the final detailed model of the system. Throughout the design process, formal verification and simulation evaluate safety, security, and performance of the system.

Keywords: Embedded Systems, ProVerif, Formal Verification

1 Introduction

To prevent the compromise of connected objects, their security vulnerabilities should be detected and mitigated, preferably as early as possible. Correcting these security flaws might be difficult once the system has been released - and sometimes impossible - if the flaws cannot be corrected by software update only. Furthermore, adding security mechanisms at the late stages of software development may change the performance of the system to render a selected architecture non-optimal.

Autonomous drones have been proposed for use in disaster relief efforts. However, insufficient security may allow an unauthorized attacker to gain control of the drone. Furthermore, disaster relief drones may carry sensitive data and images that should be kept confidential [15].

The SysML-Sec methodology was introduced to handle the design of such complex systems, in terms of safety, performance, and security [2]. SysML-Sec is an extension of UML for the design of embedded systems. It addresses system

development starting from Requirements and Attacks analysis, progressing into Hardware/Software Partitioning, and finishing with Software/System Design. The entire design process is supported by TTool, a free, open-source, multi-profile toolkit [3].

The paper presents how security can be efficiently handled through the entire design process (see Figure 1). Solid lines in the methodology represent manual steps to be performed by the designer, while dotted lines represent automatic steps performed by our toolkit. Our methodology starts with Requirements/Analysis phase, described in section 2, which helps a designer consider the security requirements and possible attacks. Next, we describe the security-aware HW/SW Partitioning phase, where we model the abstract system architecture and behavior based on those requirements and risks. Section 4 describes the transition to the final Software Design phase. Next, we present the related work in Section 5. Finally, Section 6 concludes the paper.

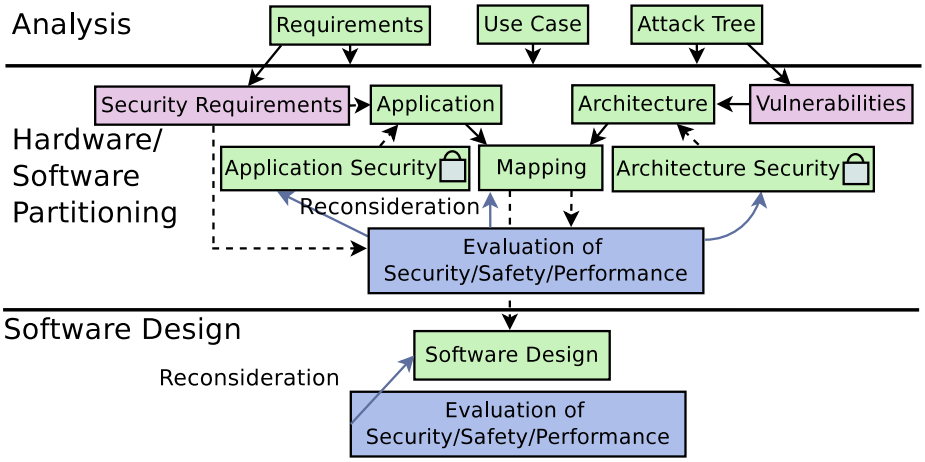

Fig. 1. Design Methodology

2 Analysis: Security Requirements and Attack Trees

The methodology depicted in Figure 1 starts with the Analysis phase (Step 1), which involves describing the requirements and use case of the system, and then considers the possible attacks that the system may face. This analysis is expected to prepare the verification phase, and to drive the two further steps (functional view, architecture view) with the selection of the components (functions, hardware elements) adapted to counter the listed attacks. The necessary iterations between requirements, attacks, and components of the system are not detailed in this paper, but are explained in [14].

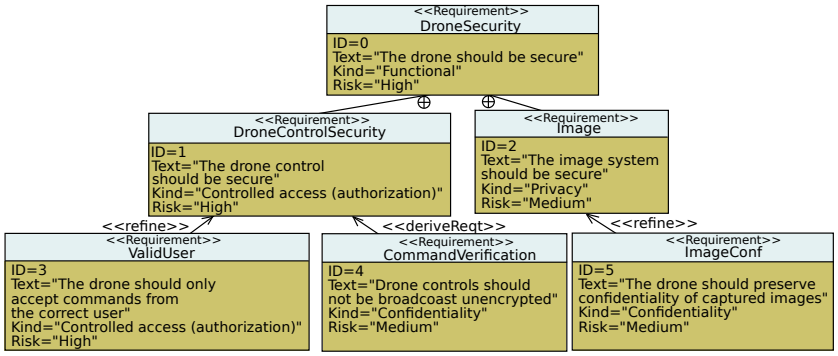

Fig. 2. Security-related Requirements Diagram for a drone

Figure 3 shows a high-level overview of one attack tree for gaining control of the drone. An attacker must understand how to forge commands, then gain remote access to the drone while denying access to the legitimate user controller. The avenues for gaining control include kicking the legitimate user and then connecting to the drone [6], or a Man-in-the-Middle (MITM) attack [13].

The Requirements Diagram includes the textual specifications regarding important properties of the system. Figure 2 shows an extract of the Security-related Requirements Diagram. These requirements may be continually refined with the details of their implementation. The requirement for the drone to be secure involves the sub-requirements that the drone should only accept commands from the authorized user, and that captured images should remain confidential.

After performing this analysis, we detail in the next section how to use these diagrams for security modeling in the HW/SW Partitioning phase.

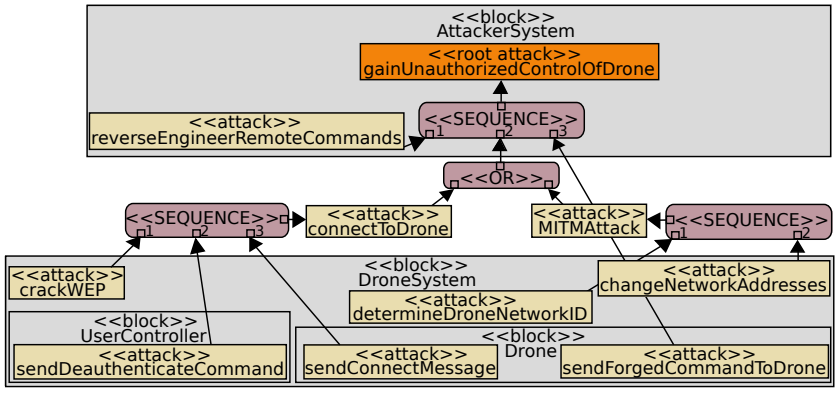

Fig. 3. Attack Tree for gaining unauthorized control of a drone

3 Security-Aware HW/SW Partitioning

The HW/SW Partitioning phases models the high-level behavior and architecture of the drone system, based on the Y-chart methodology. The application model (Figure 4) designs the high-level behavior of the system as a set of communicating tasks. The architecture (Figure 5) is modeled as the execution nodes (CPUs and Hardware Accelerators) and memories connected by buses and bridges. The mapping model then places the application tasks onto execution nodes of the architecture.

3.1 Security Modeling

Based on the security requirements described in the Analysis Phase, certain communications may be considered critical and must be secured. For example, since images taken by the camera and should be secure, we mark those communication channels with a grey lock to indicate that we should examine its security properties.

Attack Trees describing scenarios of attack also provide information on the attacker's actions and capabilities. In our attack tree, we assumed that the attacker could intercept the Wifi communications between the controller and Drone. On the other hand, since we assume the attacker has no physical access to the drone and cannot probe the internal drone bus, then we mark it as secure with the green shield.

In this sample mapping, mission commands are broadcast across a bus accessible to the attacker, so we must secure that communication.

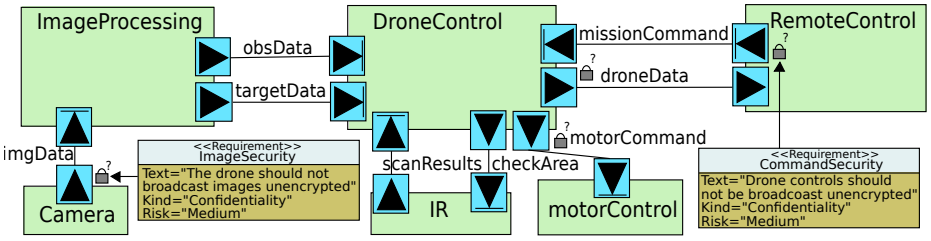

Fig. 4. Application Model of Drone considering requirements

Abstract operators named 'Cryptographic Configurations' indicate security operations performed on communications [9]. These application security elements may be added manually or automatically after a security verification. Since security operations can be computationally-intensive and require the secure storage of cryptographic elements (e.g., keys), there exist specialized coprocessors Hardware Security Modules which perform cryptographic operations faster than a normal processor and store cryptokeys.

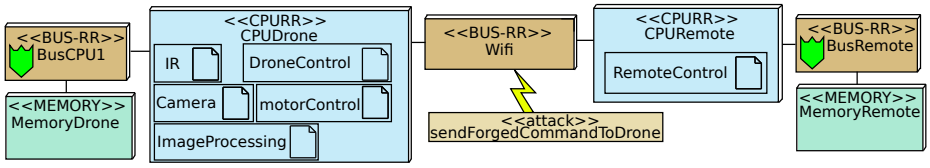

Fig. 5. Architecture Model of Drone considering attackss

To verify that these countermeasures are sufficient, our toolkit performs security verification automatically with ProVerif [12]. The verification results are backtraced automatically to our diagrams, by marking communications that are verified secure for a given mapping with a green lock, and marking insecure communications with a red lock. Once our mapping has been verified to meet all safety, security, and performance requirements, then the software of the system can be designed in detail, as described in the next section.

4 Security in Software Design

After HW/SW Partitioning has determined the architecture and mapping of the system, we design the software components of the application in greater detail. Our toolkit can generate a software design diagram automatically based on HW/SW Partitioning models. The details of the algorithms to be implemented must then be added by the designer. For example, the trajectory calculation algorithm should be added in place of the element indicating only its computational complexity. The generation saves the designer time, by creating the model blocks and framework.

Figure 6 shows the transition from the Activity Diagram in HW/SW Partitioning to the State Machine Diagram in Software Design for the drone controller. Cryptographic configurations are translated into software methods. The software modeling environment also offers primitives to closely model security protocols.

Verification of Software Design models is intended to verify the details of implementations of security. For example, the verification may concern the confidentiality of one given block attribute, or the authenticity of one given message exchnaged between SysML block. On the contrary, verification at mapping stage concerns abstract data channel.

5 RELATED WORK

[4] relies on Architecture Analysis and Design Language (AADL) models to consider architectural mapping during security verification. The authors note that a system must be secure on multiple levels: software applications must exchange data in a secure manner, and also execute on a secure memory space and

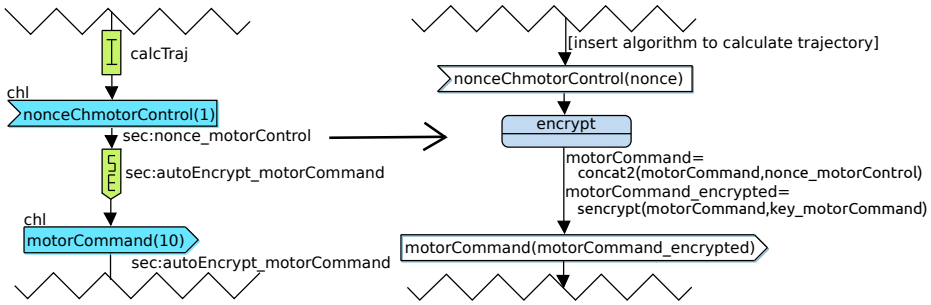

Fig. 6. Translation of Activity Diagrams to State Machine Diagrams

communicate over a secure channel. Our approach, however, considers security regarding protection against external attackers instead of access control.

Another approach performs Design Space Exploration on a vehicular network protecting against replay and masquerade attacks [10]. The project evaluates possible security mechanisms, their effects on message sizes, and candidate architectures during the mapping phase. While their work targets automotive systems and network communications, our analysis may be applied more broadly for any embedded system.

Attack Defense Trees [7] analyze the possible attacks against a system, in conjunction with the defenses that the system may implement. The supporting toolkit ADTool analyzes attack scenarios to determine the cost, probability, time, etc, required for a successful attack.

The Knowledge Acquisition in Automated Specifications approach Security Extension aims to identify security requirements for software systems [8]. The methodology uses a goal-oriented framework and builds a model of the system, and then an anti-model which describes possible attacks on the system. Both models are incrementally developed: threat trees are derived from the anti-model and the system model adds security countermeasures to protect against the attacks described in the anti-model.

SecureUML enabled the design and analysis of secure systems by adding mechanisms to model role-based access control [11]. Authorization constraints are expressed in Object Constraint Language (OCL) for formal verification. Our security model focuses on protecting against an external attacker instead of access control. In contrast to formula-based constraints or queries, our approach to security analysis relies on graphically annotating the security properties to query within the model.

Another work [16] proposed modeling security in embedded systems with attack graphs to determine the probability that data assets could be compromised. While their approach is also UML-based, they focus on estimating probabilities of success for attacks, while ours focuses on verifying adequate placement of encryption.

UMLSec [5] is a UML profile for expressing security concepts, such as encryption mechanisms and attack scenarios. It provides a modeling framework to define security properties of software components and of their composition within a UML framework. It also features a rather complete framework addressing various stages of model-driven secure software engineering from the specification of security requirements to tests, including logic-based formal verification regarding the composition of software components. However, UMLSec does not take into account the HW/SW Partitioning phase necessary for the design of IoTs.

The Software Architecture Modeling (SAM) framework [1] aims to bridge the gap between informal security requirements and their formal representation and verification. SAM uses formal and informal security techniques to accomplish defined goals and mitigate flaws. SAM relies on a well established toolkit - SMV - and considers a threat model, but the "security properties to proof" process is not yet automated. In contrast, our work focuses on automatic formal verification from an abstract partitioning model.

In contrast to these approaches, our work involves a methodology for the modeling and analysis of security at all stages in the design process.

6 Conclusion

This paper presented how our enhanced SysML-Sec Methodology now considers security at all phases in the design process. We examined the security considerations in the design of a disaster relief drone, which must not be compromised or controlled by an attacker. First, the requirements and attacks phase helps us decide what data needs to be secure and which architectural locations are vulnerable, which then leads us to add abstract representations of security. Once an architecture and mapping are decided, then we can generate the base structure of the software models. The software model is then refined to include the algorithms and details of the software to be developed.

In future work, we plan to better connect the Analysis and Partitioning phases. Currently, the Analysis phase provides guidelines for the designer, but does not explicitly connect security requirements in the Requirements Diagram with the critical channels in the Mapping Models. We should add the capabilities to trace the fulfilment of each requirement. The attack paths in the Attack Trees could also provide more explicit information regarding the security of specific architectural elements, instead of needing to be deciphered by a designer. These additions will enhance our toolkit to better support the design of secure embedded systems.

Acknowledgment

This work was partly funded by the French Government (National Research Agency, ANR) through the Investments for the Future Program reference #ANR-11-LABX-0031-01 and Institut VEDECOM.

References

1. Ali, Y., El-Kassas, S., Mahmoud, M.: A rigorous methodology for security architecture modeling and verification. In: Proceedings of the 42nd Hawaii International Conference on System Sciences. vol. 978-0-7695-3450-3/09. IEEE (2009)
2. Apvrille, L., Roudier, Y.: SysML-Sec: A Model Driven Approach for Designing Safe and Secure Systems. In: 3rd International Conference on Model-Driven Engineering and Software Development, Special session on Security and Privacy in Model Based Engineering. SCITEPRESS Digital Library, France (Feb 2015)
3. Apvrille, L.: TTool. ttool.telecom-paristech.fr (Dec 2003), `ttool.telecom-paristech.fr`
4. Hansson, J., Wrage, L., Feiler, P.H., Morley, J., Lewis, B., Hugues, J.: Architectural Modeling to Verify Security and Nonfunctional Behavior. IEEE Security Privacy 8(1), 43–49 (Jan 2010)
5. Jürjens, J.: UMLsec: Extending UML for Secure Systems Development. In: Proceedings of the 5th International Conference on The Unified Modeling Language. pp. 412–425. UML '02, Springer-Verlag, London, UK, UK (2002), `http://dl.acm.org/citation.cfm?id=647246.719625`
6. Kamkar, S.: Skyjack: Autonomous drone hacking (2003), `http://www.samy.pl/skyjack/`
7. Kordy, B., Kordy, P., Mauw, S., Schweitzer, P.: Adtool: Security analysis with attackdefense trees. In: Joshi, K., Siegle, M., Stoelinga, M., DArgenio, P. (eds.) Quantitative Evaluation of Systems, Lecture Notes in Computer Science, vol. 8054, pp. 173–176. Springer Berlin Heidelberg (2013)
8. van Lamsweerde, A.: Elaborating Security Requirements by Construction of Intentional Anti-Models. In Proc. of the 26th International Conference on Software Engineering , ICSE '04 pp. 148–157 (2004)
9. Li, L.W., Lugou, F., Apvrille, L.: Security-Aware Modeling and Analysis for HW/SW Partitioning. In: Conferénce on Model-Driven Engineering and Software Development (Modelsward'2017). Porto, Portugal (Feb 2017)
10. Lin, C.W., Zheng, B., Zhu, Q., Sangiovanni-Vincentelli, A.: Security-Aware Design Methodology and Optimization for Automotive Systems. ACM Transactions on Design Automation of Electronic Systems (TODAES) 21(1), 18 (2015)
11. Lodderstedt, T., Basin, D.A., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Proceedings of the 5th International Conference on The Unified Modeling Language. pp. 426–441. UML'02, Springer-Verlag, London, UK, UK (2002), `http://dl.acm.org/citation.cfm?id=647246.719477`
12. Lugou, F., Li, L.W., Apvrille, L., Ameur-Boulifa, R.: SysML Models and Model Transformation for Security. In: Conferénce on Model-Driven Engineering and Software Development (Modelsward'2016). Rome, Italy (Feb 2016)
13. Rodday, N.: Hacking a Professional Drone. Slides at www.blackhat.com/docs/asia-16/materials/asia-16-Rodday-Hacking-A-Professional-Drone.pdf (Mar 2016)
14. Roudier, Y., Idrees, M.S., Apvrille, L.: Towards the Model-Driven Engineering of Security Requirements for Embedded Systems. In: proceedings of MoDRE'13, Rio de Janeiro, Brazil (Jul 2013)
15. Tanzi, T.J., Sebastien, O., Rizza, C.: Designing Autonomous Crawling Equipment to Detect Personal Connected Devices and Support Rescue Operations: Technical and Societal Concerns. The Radio Science Bulletin 355(355), 35–44 (2015)
16. Vasilevskaya, M., Nadjm-Tehrani, S.: Quantifying Risks to Data Assets Using Formal Metrics in Embedded System Design, pp. 347–361. Springer International Publishing, Cham (2015), `http://dx.doi.org/10.1007/978-3-319-24255-2_25`