Model-guided Security Analysis of Interconnected Embedded Systems

Yasamin Mahmoodi, Sebastian Reiter, Alexander Viehl,
Oliver Bringmann and Wolfgang Rosenstiel
FZI Forschungszentrum Informatik, Haid-und-Neu-Str. 10-14, D-76131 Karlsruhe, Germany

Keywords: Security Analysis, Virtual Prototyping, UML Profile.

Abstract: Software-intensive and networked embedded systems implement more and more security critical tasks. The following paper presents a framework to support security analysis along the design process using virtual prototypes (VPs). VPs describe the interconnection between different system components, include actual application codes and even integrate existing physical prototypes. These enable the user to detect structural security flaws, implementation flaws and even hardware-based security problems. Benefits of using VPs are the early availability in the design process and the fact that VPs are based on software, therefore established security analysis methods for software can be applied. This paper provides a methodology and tooling support to apply VP in the context of security analyses. Especially the integration in a model-driven design (MDD) process is highlighted. A proposed security UML profile as well as code generation ease the VP-based analysis.

1 INTRODUCTION

Embedded systems are increasingly networked by information and communication technologies (ICT). This offers a high benefit in application domains such as automotive or industrial automation. At the same time, it creates new challenges, such as the increasing complexity for security risk assessment. Car2X technologies are an illustrative example from the automotive domain. They exchange sensor and context data between vehicles and the traffic infrastructure to increase resource efficiency, safety and comfort. However, in addition to the outstanding advantages, comprehensive networking also results in new security risks. Without sufficient security measures, individual, manipulated messages are sufficient to affect the correct behaviour of Car2X-based assistance systems, such as demonstrated by (Charlie and Chris, 2015). In an extreme case, critical safety-relevant system components are manipulated, resulting in an unacceptable threat to people or assets. Unauthorized access to system-internal controls is possible by poorly secured radio interfaces such as Bluetooth, Wi-Fi, keyless entry or tire pressure systems (Checkoway et al., 2011). "Security by Design" is therefore required, to get the full benefit of comprehensively networked systems. This requires a comprehensive consideration of security measures along the design process.

We propose an approach to enable security analysis along the design process of software intensive and interconnected embedded systems. The approach provides a comprehensive and executable specification of the system architecture as well as mechanisms, procedures and parameters of security measures in the form of a virtual prototype (VP). A VP represents the preliminary stage of a physical prototype in the design process and denotes the complete or partial provision of system sub-components as executable models. Different security analyses such as penetration testing or structural analyses, including dynamic data flow analyses, are based on the VP. Virtual Prototyping is widely used in the field of electronic system design to analyze components and systems for their functional and non-functional properties (Reiter et al., 2016). The accompanying use of VPs in the development of embedded systems makes the complexity of the design of secure embedded systems manageable.

A tight integration of the VP in the design flow and a comprehensive tool support, such as automated code generation are required to foster the usage of VPs. In this paper, we highlight a modeling approach used to guide the security analysis with VPs thus reducing manual overhead. We use the Unified Modeling Language (UML) (OMG, 2011b) as modeling language. The goal is to have a well-defined, easy-to-handle user interface to manage the analysis and document the system's security features. In addition, the model eases manual analyses, such as penetration testing, by enhancing the documentation and highlighting potential design flaws.

Mahmoodi, Y., Reiter, S., Viehl, A., Bringmann, O. and Rosenstiel, W.
Model-guided Security Analysis of Interconnected Embedded Systems.
DOI: 10.5220/0006724606020609
In *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018)*, pages 602-609
ISBN: 978-989-758-283-7

The paper is structured as follows. First, the envisioned methodology is highlighted in Section 2. Section 3 shortly reviews modeling approaches used for safety analysis. Section 4 introduces the developed security profile. In addition, a comparison of the developed profile with a state of the art UML profile as well as an exemplary model is presented. Section 5 highlights the used VP framework and demonstrates the security profile integration. The paper is concluded in Section 6.

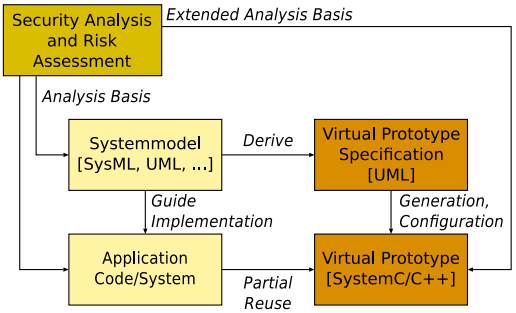

Figure 1: Design support by virtual prototyping.

2 METHODOLOGY

As motivated in the introduction, the VP is an executable simulation model of the physical system. The technique of virtual prototyping enables the evaluation of hardware/software systems without the need of physical prototypes. VPs are often used during software development, if the target hardware is not available yet. A VP simulates the required system components. Depending on the amount of simulated system parts and the level of detail, different analysis goals can be pursued. With a complete system simulation on a very abstract level, a rough evaluation, such as a structural analysis that analyses the data flow between components, is possible. Integrating actual software implementations within the VP, enables the evaluation of the target software. This step often involves a refinement of the VP, e.g. to simulate the accessed registers. Enabling the execution of a rough evaluation in an early design phase with a less detailed system model and a continuous gain in accuracy by refining the system model. The presented approach uses the event-driven SystemC (IEEE Computer Society, 2011) simulation language. SystemC covers different abstraction levels of hardware and software systems and enables both the modeling of SW applications as well as digital/analog electronic components. SystemC provides abstract Transaction Level Models (TLM) to specify the interaction of system components and enables a very early structural analyses of the complete system.

As SystemC is a C++ class library it is possible to integrate actual application code that is compatible with C++. This reduces the overhead required to create the VP and enables the analysis of the actual implementation. Flaws can be found that would lead to security vulnerabilities. Figure 1 sketches the proposed development flow. The overall goal is to execute security analyses combined with risk assessment. Traditional design flows often use physical prototypes, i.e. when the application code and the system hardware are available. Today's system design already benefits from a set of models. This models include the specification of the system architecture with SysML (OMG, 2012) as well as dedicated software aspects or use cases with UML. Only few security analysis approaches are based on these early models. The Cyber Security Modeling Language (CySeMoL) (Sommestad et al., 2013) analyzes a range of different attack types and security measures based on structural models. The approach is based on probabilities for the effectiveness of security measures. These probabilities are typically determined with the final system or based on experiences with previous systems. Therefore, it is not possible to give an analysis that can be refined along the design process.

Our approach introduces an UML model of the VP. The idea is to reuse already existing models from the system design. Compatible models such as SysML- or UML-based models can be reused by Model-to-Model (M2M) transformations. We decided to use a separate UML model, because the VP specification may contain additional information that should not pollute the actual system model. An example of such pollutions are functions that model timing in the VP or functions to read data from the test bench. Based on this UML-based specification we generate the structural aspects of the VP as well as the simulation runtime configuration. For detailed information about the provided automation, see Section 5.

Another aspect that is highlighted in Figure 1 is that the VP can integrate actual application codes. The VP is a C++ implementation, therefore source codes are directly integrated and compiled. Binary codes are integrated with emulators, such as QEMU or with Instruction Set Simulators (ISS). The use of existing software reduces the overhead to create the VP and makes the simulation more accurate.

This paper focuses on the UML-based security modeling based on UML models of the VP. Before highlighting our approach, different existing security modeling approaches are evaluated.

3 EXISTING MODELING APPROACHES

Several approaches exist for security assessment and modeling. UMLsec is an extension to the Unified Modeling Language for integrating security related information in UML specifications. The central idea of the UMLsec extension is to define labels for UML model elements (stereotypes), that add security-relevant information to these model elements (Jürjens, 2010; Jürjens, 2002) when attached. The approach is strongly influenced by networked information systems in business infrastructures, such as distributed information systems (Best et al., 2007) or electronic payment services. The extension contains for example stereotypes for «`fair exchange`», specifying the requirement of a fair trade exchange, or «`Internet`», «`LAN`» to model links between participants. Some of the presented stereotypes apply for networked embedded systems, e.g., «`integrity`» or «`encrypted`». Others are too specialized to cover all aspects of embedded systems, such as «`Internet`» or «`LAN`». These two stereotypes are too specialized to cover the different bus systems available in a vehicle. Others are not required in the context of automotive systems, such as «`fair exchange`».

SecureUML on the other hand provides an UML extension for specifying role-based access control (RBAC) and other access control policies (Basin et al., 2006; Lodderstedt et al., 2002). SecureUML mainly focuses on UML class diagrams and does not provide stereotypes for other UML diagrams. Based on its focus on RBAC, the profile does not provide the required generality for security modeling, e.g., to specify the attack surface or protection goals. In addition, only class diagrams are supported, a structural system specification is not considered.

The Cyber Security Modeling Language (CySeMoL) is another approach with the purpose of modeling attacks and assessing the attack impact. CySeMoL analyzes a range of attack types and security measures. CySeMoL's output is probabilistic and estimates the probability that different attacks can be accomplished against assets in the system architecture (Sommestad et al., 2013). The listed attacks provide a good overview but the language do not cover all aspects that are required to model security concepts on abstract system level.

Another examined approach is Secure Tropos. It is an extension of Tropos used to model and analyze security requirements alongside functional requirements. The methodology provides a requirement analysis process that supports the designer from the acquisition of requirements to their verification (Bresciani et al., 2004). The SI* modeling language evolved from the I* modeling language that employs the notation of actor, goal, task, resource and social dependency between actors to represent the functional design of the system (Massacci et al., 2010). While the language offers some interesting extensions, it was not suitable for lightweight annotation, required for the VP specification.

Each of the examined approaches has been defined for certain application areas and has its benefits and drawbacks. However, in our view, none of them is adequate to cover all requirements to specify the system security concepts in detail, required to configure a VP-based security analysis. Therefore, we were convinced to define our own security modeling approach presented in this paper.

4 SECURITY UML PROFILE

The model-based virtual prototype specification (VPSpec) provides models for the required system components and their assembly to a simulation instance. The VPSpec therefor automatically covers the security concepts of the system, such as separation, the inherent weak points of the system architecture or the protection goals of the system. Additional information such as the attack surface or the protection goals is missing, for an security analysis. Other information is inherently specified in the VPSpec, such as the communication protocol implemented by a component. Therefore, some additional information is added that is only used for documentation of the analysis. The center of the VP-based security analysis is the VPSpec that is extended with this security related information. It offers the user a single point of information and different tasks are automated thus reducing the manual overhead. Figure 2 depicts the envisioned structure. In the center is the UML-based VPSpec that is derived from the actual system model (compare Figure 1). The model is mainly used to generate and configure the VP, both with injectors to simulate the attack surface and monitors for the protection goals. In addition, the model is used for static data flow analysis and for a general documentation of the system's

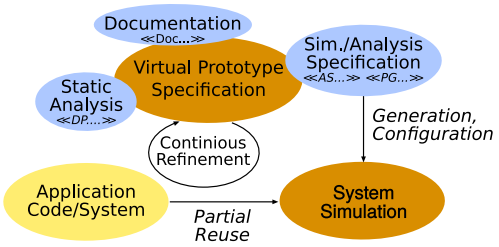

Figure 2: Application of the UML security profile.

security features. Both the VP and the VPSpec are refined during the design process. UML profiles are used to specialize the UML for the VPSpec.

Information security is defined as protection of information and information systems against unauthorized users to ensure integrity, confidentiality and availability. Integrity refers to maintenance or assurance of accuracy and trustworthiness of these data over its life cycle. Confidentiality means that unauthorized users do not have the chance to access the data, processes or entities. On the other hand, availability ensures a reliable access to information in time. We define our security profile along the triangle of confidentiality, integrity and availability.

Analyzing a system from the security point of view, weak input points and vulnerable components of the system should be specified. Automotive systems benefit from remote connections for Remote Key Entry Systems or internet access. Every connection can bring new opening for attackers and lead to malicious activities.

Attack Surface

The attack surface is the total vulnerability and shows weak points of the system, where an unauthorized user can access, change or insert invalid data. Attack surfaces can be categorized as network attack surfaces, software attack surfaces and physical attack surfaces. Attack surfaces of the system should be specified in the model-based VPSpec.

In our security profile, we define five stereotypes to represent vulnerable points and to define the amount of effort, which is needed to break down vulnerable objects. The effort is added as an abstract value that is associated with the stereotype. The stereotype, «`As.DataRead`», describes the possibility of reading the data by an unauthorized user. It also defines the level of effort, which is needed to read the data. «`AS.DataModify`» is another stereotype, which states if an attacker can change the data of a property and how much effort is needed. The stereotype «`AS.DataDelete`» specifies how much effort is needed for an unauthorized attacker to delete the data of the stereotyped property. «`AS.DataAdd`» is another stereotype which specifies the possibility of adding data by an unauthorized user and the level of effort needed. At the end, «`AS.ServiceUnavailable`» specifies how much effort an attacker needs to make an operation unavailable in the system. The presented stereotypes are used to specify the attack surface within the VPSpec. This information will have positive effects on the placement of attack points, e.g., with injectors, in the VP.

Protection Goal

Embedded systems contain data and components with different levels of criticality. With the focus on automotive systems, an attacker may hack the head unit and try to control the volume of the tuner changing the HVAC temperature or other malicious activities to disturb passengers. With more effort, the attacker may access the critical data and be able to send CAN packets to control the brakes, which will lead to serious consequences. Our extended VPSpec enables the specification of critical parts of the system which require protection. The model contains additional information about the severity level of these protection goals. Five stereotypes in our meta-model are devoted to these protection goals to show critical parts of the system. The stereotype «`PG.DataConfidential`» defines that the associated property should be protected against reading or predicting by an attacker. Using the tag *Severity*, it also determines the level of severity if the protection goal is violated. The second stereotype of this category, «`PG.DataModify`» indicates that the property should be protected against modification; it also defines the level of severity in case of violation. «`PG.DataDelete`» specifies that it should not be possible for an unauthorized user to delete this information. Similar to the other stereotypes a level of severity is added. The stereotype «`PG.DataAdd`» indicates with its severity that this property should be protected against adding new data by an attacker. The last stereotype «`PG.ServiceAvailability`» indicates that the stereotyped operation must be available and should not be stopped. With this information, the user can plan suitable attack scenarios. Additionally, monitors can be added to evaluate if an injected modification leads to a violation of the protection goal, at least for the protection goals w.r.t. modification.

Documentation

To make the documentation-related information visible to the user, different information is explicitly modeled in the VPSpec. General security information such as handshaking protocols will be provided with the stereotype «`Doc.SecurityProtocol`». Encryption protocols are abstract protocols which give information about the cryptographic methods. A weak encryption may lead to vulnerabilities and bring new openings to attackers. Information about applied cryptographic methods is necessary for security analysis, which in our model is provided by stereotype «`Doc.EncryptionProtocol`». Security communities regularly publish known vulnerabilities, exposures and related security patches. Appending more details about the VP components such as the software version enable finding vulnerable points based on published vulnerabilities and exposures. The VPSpec

encloses software functionality with an UML class. Therefore such information would not be visible. We devoted one stereotype, «Doc.SoftwareVersion», e.g., to specify software versions for comparison with known security issues. Authorization is an access control method which specifies access rights for resources. Information about authentications gives a better view about security measure of the system. Stereotype «Doc.Authorization» indicates that a client who requests a connection to the stereotyped operation first needs to be authorized. Authentication is a security mechanism for confirming the identity of data or users. By adding authentication information in form of the stereotype «Doc.Authorization», our model will have a more comprehensive view. Summarising, many information is inherently modeled in the VP. By selecting a component that implements encryption or decryption, the functionality is automatically regarded in the VP. We added these stereotypes to make this information more explicit in the UML model.

Data Flow

Data flow graphs and control flow graphs of the system make it possible to analyze data spread in the system and detect the accessibility of critical regions. Our security profile offers one stereotype named «DP.DataFlowElement» which provides information about the data flow in the system and could be used for static analysis. The idea is to use UML state machines to visualize data dependencies and analyze how the protection goals and the attack surface are linked. We currently evaluate two approaches to derive this graph: static code analysis and dynamic taint propagation based on the VP.

We should sum up that our security profile specifies confidentiality and integrity aspects related to the attack surface and the protection goals. Stereotypes «AS.ServiceUnavailable» and «PG.ServiceAvailability» on the other hand, specifies requirements for the availability of the system. The profile is defined using stereotypes, tag definitions and constraints, which are applied to specific model elements, like Classes, Attributes, Operations, and Activities. Table 1 indicates the stereotypes of the proposed security profile which we described above. The stereotypes are associated to the UML-based VP-Spec instead of the other models. We support different M2M-transformation, based on QVTo (OMG, 2011a), for our approach. This way the stereotypes are indirectly associated with the other models.

Table 1: Extract of the security profile.

Stereotype	Base class	Tag
AS.DataRead	Property	Effort
AS.DataModify	Property	Effort
AS.DataDelete	Property	Effort
AS.DataAdd	Property	Effort
AS.ServiceUnavailable	Operation	Effort
PG.DataConfidential	Property	Severity
PG.DataModify	Property	Severity
PG.DataDelete	Property	Severity
PG.DataAdd	Property	Severity
PG.ServiceAvailability	Operation	ResTime
Doc.EncryptionProtocol	Class	String
Doc.SecurityProtocol	Class	String
Doc.SoftwareVersion	Class	String
Doc.Authentication	Operation	
Doc.Authorization	Operation	
DP.DataFlowElement	State	

4.1 Comparison

In the following section, we compare our envisioned profile with the exiting profile UMLsec. As the focus on UMLsec lies more on enterprise security it contains stereotypes that are not applicable to embedded systems. Nethertheless, it offers general security stereotypes and in this section it should be evaluated if the envisioned profile covers the essential subset of UMLsec.

UMLsec offers stereotypes like «fair exchange» and «provable» originating from e-commerce scenarios. They specify the requirement of a fair product exchange, e.g. with the help of a trusted third party or of provable payments. Considering the focus of our model to the embedded systems, we did not devote stereotypes for this aspect. However, it should be mentioned that the functionality of these mechanisms is modeled implicitly by the VP, e.g., by providing simulation modules that implement a provable payment.

We broke down the role-based access to the involved mechanisms by using the stereotypes «Doc.Authentication» and «Doc.Authorization» to substitute the UMLsec stereotype «rbac».

UMLsec uses the stereotypes «Internet», «LAN» and «wire» to model connections with different attack possibilities. Our proposed profile provides the stereotypes «AS.DataRead», «AS.DataModify», «AS.DataDelete» and «AS.DataAdd» to characterize the attack surface of different connections. Furthermore, encrypted connections which are mod-

eled with the stereotype «encrypted» in UMLsec will be specified in our profile by the combination of «AS.DataRead» and «Doc.Encryption». The stereotypes «smart card», «POS device» and «issue node» in UMLsec are nodes with varying protection mechanism. Our stereotypes to specify the attack surface are used to specify the characteristics of the different devices.

The assumption of secrecy and integrity as well as high sensitivity is indicated with the stereotypes «secrecy», «integrity» and «high» in UMLsec. In our model, stereotypes for protection goals can model the secrecy, integrity and level of sensitivity. The concepts of the stereotypes «critical», «secure links», «data security», « no down-flow » and «no up-flow» are modeled with the help of the protection goal stereotypes.

The stereotype «Doc.Authorization» specifies a property which is protected by an authorization mechanism. This mechanism ensures that the object can be accessed only through authorized parties. This is a proper stereotype to model guarded objects which can be accessed only through their guard. In UMLsec, this concept is model with the stereotypes «guarded» and «guarded access». «secure dependencies» is a stereotype of UMLsec which ensures secure dependencies. Secure dependencies are covered with the stereotypes «Doc.Authorization» and «Doc.Authentication».

In our opinion, all relevant aspect of UMLsec can be modeled with our stereotypes on a more fine grained level, what is necessary for considering embedded security.

4.2 Example

Hacking modern automotive embedded systems often requires three steps. First, the attacker needs to gain remote access to the internal network of the vehicle. This step is usually accomplished by sending wireless signals and compromising the control unit in the front line. This step leads the attacker to the next step, which is injecting messages to the car network and directly or indirectly controlling desired electronic control units (ECUs). Here it should be considered that in many designs, it is not possible to send a message to the safety critical ECUs directly. Therefore, the attacker has to bypass gateway ECUs to forward the compromised message to the critical ECUs. The last step will be forcing the target ECU to compromise safety features. The whole attack needs information on messages and structure of the automotive network.

Our security profile covers all steps of the attack by its stereotypes. The stereotype regarding the attack

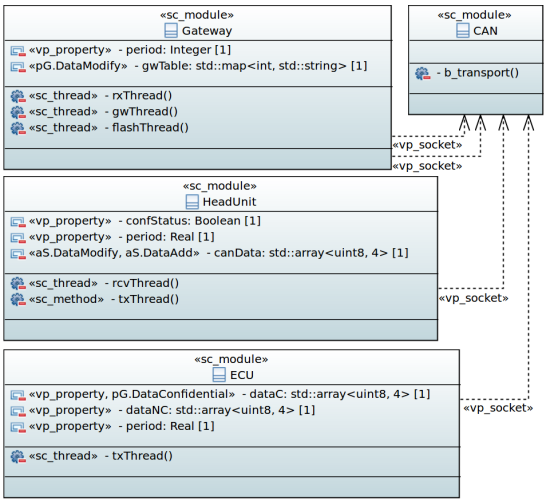

Figure 3: VPSpec of the simulation component.

surface will mark penetration possibilities. All the attack points of the various steps should be marked. That means, not only the first attack surface only, but also it is assumed that an ECU is compromised, an extended attack surface is possible. This allows to evaluate isolated attack steps with the VP. The next information that is specified in the model are the protected safety critical ECUs or gateway ECUs. This is done with the protection goal stereotypes. Finally, documentation-related stereotypes offer information about the applied security mechanisms or software versions to estimate vulnerabilities and guide the manual penetration tests.

Here we focus on an attack scenario based on the aqLink protocol within the Gateway which controls both voice and data cellular communication. First, an attacker employs some flaws in the authentication procedure. In a second attack, the attacker uploads an exploit to the telematics system. The attacker hacks the head unit and then sends compromised messages to the safety critical ECUs by spoofing messages. Before the critical ECU can be reached, the filtering mechanism of the gateway has to be deactivated. Figure 3 models components involved in the attack to compromise the gateway. It shows three components that represent the functionality of the different ECUs. The fourth component is the field bus used for communication. In the system structure two instances of the CAN bus are present; one with sensitive information (Body-CAN) and one with non-critical signals (Comfort-CAN). The used structure is shown in Figure 5. The chosen abstraction level is TLM, because the model is mainly used for a structural data flow analysis. Some information stored in the head unit is assumed readable by an attacker. In this scenario, it is assumed an attacker

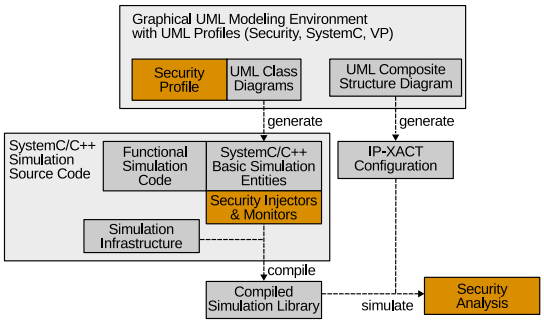

Figure 4: Specification and analysis flow.

has already compromised the head unit. This information is marked as «AS.DataRead». On the other hand, an internal ECU that is connected to the second bus contains information that should not be read by an attacker. Therefore, this information is stereotyped as «PG.DataConfidentiality». The goal of this abstract VP is to detect security flaws within the gateway, such as an unauthorized flash mechanism to change the routing table. In the original scenario, it was possible to configure the gateway over CAN. The configuration included the writing of a filter that should prevent sensitive information to be forwarded on the public CAN. The data flow analysis showed that both the filter condition and the forwarded data where dependent on the same input data, in this case the CAN message payload. This dependency showed that if the input data is compromised the filter is jimmied. This very basic example showed how an executable specification could be used to analyze the data propagation paths between the attack surface and the protection goals to find potential weak spots. The analysis is not yet fully automated. It gives only hints to the user to take into account, such as this concentration point at the filter. By adding e.g., an authentication mechanism for overwriting the filter, the data flow analysis would show the same result. But the authentication would prevent the attacker from changing the routing table. Therefore, our approach highlights a false positive, under the assumption the authentication is not corruptible.

5 VIRTUAL PROTOTYPE FRAMEWORK

The presented approach uses a subset of the UML to specify the VP and its security analysis related information. In this section, the link between the modeling and the VP should be described. An overview is given in Figure 4.

As already highlighted, the UML-based VPSpec is basis for the security analysis. The simulation library is generated from this specification. The library contains all system components required for one simulation run. This library is compiled and for each simulation run a configuration file is given that determines the simulation instance. This configuration file is again generated from the UML-based VPSpec. The different security analyses, such as the dynamic data flow analysis or the manual penetration testing are based on the executable simulation.

Before going into detail on how the VP is linked to the VPSpec, let highlight the general procedure of our VP approach. The VP consists of the design under test (DUT) and its test bench. The DUT models software, hardware and analog system parts. It consists of a modular, parameterizable system simulation. One simulation instance is assembled of parameterizable simulation entities (SEs). To facilitate the analysis of different DUT characteristics, the framework uses a dynamic configuration that is supplied during runtime. This covers the DUT architecture as well as the SEs parameterization. This provides a high re-usability of existing simulation models.

The components of the VP are specified using UML class diagrams. Each simulation component can be specified using an UML class, attributes and operations. Dedicated stereotypes, to specify SystemC related declarations such as SystemC threads (*sc_thread*) are added with additional profiles, beside the already highlighted security profile. Based on the VPSpec automatically the structural aspects of the C++ code are generated. This covers all class definitions, member variables and member methods as well as SystemC macro definitions such as *SC_HAS_PROCESS*. The member methods are only generated as empty stubs which have to be implemented by the user. The code generation is realized, by different Java Emitter Templates (JET).

One key element of our VP-based approach is that the VP is assembled and configured by a file during execution. All VP components provide a fabric method to dynamically instantiate and configure them as well as some methods to interconnect the created instances. This way it is not necessary to program a top module that assembles and configures all components. The configuration file is an IP-XACT design specification (SPIRIT Consortium, 2009) that lists all the components to instantiate, how to parametrize them and how to interconnect them. Basis for the generation of the configuration file is an UML composite structure diagram.The diagram instantiates the already defined classes, assigns parameters and inter-

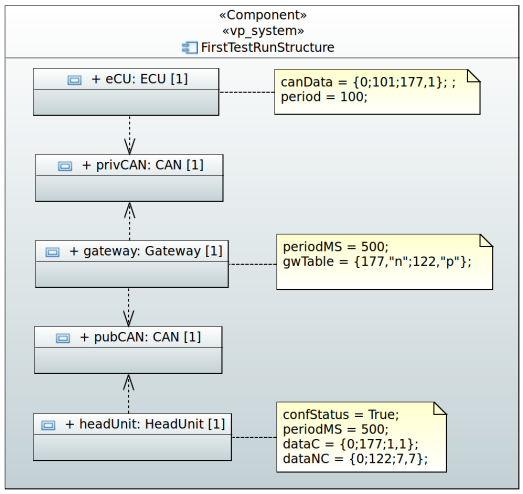

Figure 5: UML model to specify the simulation structure.

connects them. Figure 5 highlights such a specification. It instantiates the classes from Figure 3 and interconnects them. With such a specification, the configuration file is generated and the simulation is executed. The example shows the structural specification of the previous example with a head unit an internal CAN (Body-CAN) and a public CAN (Comfort-CAN). Both are interconnected via a gateway that is the target of the analysed attack.

6 CONCLUSION

In this paper, we outlined our ambition towards the modeling of security concepts in the context of VP-based security analyses. Modeling is applied for the specification of the attack surface, protection goals and further documentation. With this comprehensive information base, different security analysis based on VPs are executed, such as a dynamic data propagation analysis or virtual penetration testing. Currently the available analyses are largely manual tasks e.g., the interpretation of the data propagation analysis to see if the attack surface is linked to the protection goals, without security measures. In the future, we will try to automate the tasks and reduce the manual overhead for the user. The presented modeling approach already provides various steps of automation, but these mainly support the user to generate the VP and plan the attack scenarios.

ACKNOWLEDGEMENTS

The work was partially funded by the Baden-Württemberg Stiftung gGmbH.

REFERENCES

Basin, D., Doser, J., and Lodderstedt, T. (2006). Model driven security: From uml models to access control infrastructures. *ACM Trans. Softw. Eng. Methodol.*

Best, B., Jurjens, J., and Nuseibeh, B. (2007). Model-based security engineering of distributed information systems using umlsec. In *29th International Conference on Software Engineering*.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

Charlie, M. and Chris, V. (2015). Remote exploitation of an unaltered passenger vehicle. *Black Hat USA, 2015*.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T. (2011). Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 6–6, Berkeley, CA, USA. USENIX Association.

IEEE Computer Society (2011). IEEE 1666-2011 Standard SystemC Language Reference Manual. *IEEE Std 1666-2011*.

Jürjens, J. (2002). Umlsec: Extending uml for secure systems development. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, UML '02, pages 412–425, London, UK. Springer-Verlag.

Jürjens, J. (2010). *Secure Systems Development with UML*. Springer-Verlag, Berlin, Heidelberg.

Lodderstedt, T., Basin, D., and Doser, J. (2002). *SecureUML: A UML-Based Modeling Language for Model-Driven Security*, pages 426–441. Springer Berlin Heidelberg, Berlin, Heidelberg.

Massacci, F., Mylopoulos, J., and Zannone, N. (2010). *Security Requirements Engineering: The SI* Modeling Language and the Secure Tropos Methodology*, pages 147–174. Springer, Berlin, Heidelberg.

OMG (2011a). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification.

OMG (2011b). Unified Modeling Language.

OMG (2012). OMG Systems Modeling Language (OMG SysML).

Reiter, S., Viehl, A., Bringmann, O., and Rosenstiel, W. (2016). Fault injection ecosystem for assisted safety validation of automotive systems. In *2016 IEEE Int. High Level Design Validation and Test Workshop*.

Sommestad, T., Ekstedt, M., and Holm, H. (2013). The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Systems Journal*, 7(3):363–373.

SPIRIT Consortium (2009). IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows.