Security Analysis of Embedded Systems Using Virtual Prototyping

Yasamin Mahmoodi
Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe, Germany
mahmoodi@fzi.de

Sebastian Reiter
Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe, Germany
sreiter@fzi.de

Alexsander Viehl
Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe, Germany
viehl@fzi.de

Oliver Bringmann
Universität Tübingen
Sand 13 72076 Tübingen, Germany
bringman@informatik.uni-tuebingen.de

***Abstract*—Connected embedded systems have a significant role in modern life. Prominent benefits of this new concept and how it eases our life is irrefutable. However, as this technology provides users remote connection, it opens new opportunities for attackers to get access to the system and perform malicious activities. In order to get full benefits of connected embedded systems, security should be considered during system design. As a proposed approach to produce secure embedded systems, we offer a comprehensive attitude based on security by design concept. The proposed solution is inspired by Secure System Development Life Cycle (SSDLC) models and employs virtual prototyping to present an early security evaluation during development process of embedded systems.**

***Keywords: Security analysis; IoT; Security requirement; Virtual prototyping; Security-by-design*.**

I. INTRODUCTION

Embedded systems are an essential part of modern life. Transportation systems, from automotive to airplanes, increasingly apply embedded system to elevate performance and comfort. Medical instruments, on the other hand, profit from embedded systems for monitoring applications. Smart home is another area in which embedded systems showed up to offer light and temperature controlling, pet and baby care, smart kitchen and many other applications. Connected embedded systems including connected automobiles, connected smart homes and wearable technology bring up the capability of remote monitoring and functioning.

Alongside the distinguished roll of connected embedded systems in our modern life, security challenges which may be turned up by these new technologies should be considered. Security flaws in embedded systems may lead to privacy and financial problems or endanger people's life. A vulnerability in embedded systems may give attacker the opportunity to access critical information and perform malicious activities.

In order to make a profit of outstanding advantages of connected embedded systems, security measures should be considered during design and development phases. Security by design [1] is a proper solution, which helps system designers produce the system from the ground up to be secure. In this method, security aspects of the system will be considered in the entire life cycle of system development, from system specification and modeling to system development and assembly. Corresponding to each development phase, several test and analyses will be considered. As the ultimate destination, penetration testing on the final product will take place. Therefore, security considerations should play an equal role in system development process alongside functional and usability features.

As a solution, we propose a comprehensive model-centric methodology to consider and evaluate security during the design process in the form of a Virtual Prototype (VP). The approach offers automated security consideration which enables early architectural analyses and provides preparation of penetration testing by architectural analyses, as well as searching for already known weaknesses.

In recent years, virtual prototyping has been applied in system development process in the automotive industry and other industrial products. In this area, virtual prototyping is applied as a platform for software development, architecture modeling as well as system implementation. We believe that virtual prototyping has the capability to play a significant role in security analysis and security assurance for embedded system design. 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. With VP, different security analyses such as penetration testing or structural analyses, including dynamic data flow analyses during the design and development process are possible.

The paper is structured as follows: Section II highlights previous works which focus on secure system development approaches. Section III introduces considering security during design and development phases of embedded systems development cycle. The proposed solution in order to design and develop secure embedded systems is explained in Section IV. In Section V, techniques and methods to support the proposed framework are mentioned. Finally, in Section VI, a conclusion of the paper is presented.

II. Related work

Current system development approaches, such as [2] and [3] do not consider security features in system design and development life cycle. Several works [4], [5] integrated security considerations into development life cycle; However, these works are limited to software systems and embedded systems features are still missing.

MILS (Multiple Independent Levels of Security) platform introduced in [6] is based on security by design approach which break the system into partitions and defines partition categories based on their criticality. The focus of this work is on system architecture and does not offer security consideration for all development steps. ESSAF (Embedded System Security Assessment Framework) [7] offers collaborative security assessment in embedded systems development. The focus of this work is to document the assets, threats and security mechanisms. Security evaluation of the embedded systems is divided into tree phases, System Modeling, Security Modeling and Mitigation Planning. ESSAF provides a well defined documentation approach for security evaluation of embedded systems during design and development phases. Yet, no mechanism is offered to cover hardware features during security assessment.

In [8], a secure-by-design process for cyber-physical systems is represented which integrated secure software engineering practices into a discipline spanning engineering process for cyber-physical systems. In this paper, the authors added platform-independent features to security requirements in software design processes in order to consider hardware characteristics of the system. Threat models, misuse case and anti-requirements with platform consideration are added to the existing model-based system engineering. However, lack of a comprehensive security assessment during the system development life cycle, from requirement specification to the final prototype development is visible.

[9] suggests a design methodology to integrate and evaluate security mechanisms into design process of embedded systems. In this paper security requirements are defined, then suitable security mechanisms based on the requirements will be offered and evaluated. The results of security mechanisms integration are assessed based on proper metrics. However, comprehensive evaluation during all design and development phases is not consider in this work.

To sum up, several works focus on security evaluation of embedded systems. It should be mentioned that most of them do not consider security analysis during design and development life cycle. However, a few approaches which bring security evaluation into development cycle do not offer comprehensive approaches. More important, the gap between current development state and final system prototype is noticeable in most of the works. To be specific, at each step of design and development, evaluation and tests will take place only on parts of the system which are already developed and characteristics of physical parts of the system are missing. This issue specially for embedded systems which lay on hardware environment is important.

III. Security by design

Security by design is a methodology which offers considering security features during all design and development phases of the systems, from planing phase and requirement analysis to final prototype. Secure System Development Life Cycle (SSDLC) [10] as security by design technique integrates security measures throughout design and development cycle and brings up security assessment corresponded to each development step [11]. SSDLC is in fact adapted version of traditional System Development Life Cycle (SDLC), which includes security assessment capability.

Several models for SDLC are presented and each model traces a set of phases unique to its type to ensure expected outcomes in system development process. Waterfall model, agile model, V-shaped model, iterative model, Spiral Model and big bang model [2] are some of the popular techniques for SDLC. V-shaped model and iterative model draw our attention as potential models to offer security during design and development phases of the system. V-shaped model at very first step of system design starts with planing and requirement specification and goes deeper with high level design, then low level design and the final implementation. In this model, corresponding to each design and implementation phase a set of tests is considered. On the other hand, iterative model begins with implementation of a simplified version of the system and at each iteration add more details to it until the implemented system is sophisticated enough to ensure the requirements. In this model, at the end of each iteration developed system will be tested and refined.

By addressing security measures to the SDLC pipeline, security evaluation will be presented from the outset. Proposed security analysis approach is inspired by V-shaped and iterative model to offer security by design for embedded systems. V-shaped model makes a profit of mapping development phases to test packages and iterative model suggests refinement after each iteration. Bringing both models into practice enhances security by design approach.

IV. Proposed approach

We offer an analysis process based on virtual prototyping to support comprehensive security evaluation of connected embedded systems. The process enables security assessment of the system, security mechanisms estimation and architectural evaluation of the system under development and supports the designers to make efficient design decisions during the design phases. The presented approach closes the gap between initial design and security tests on the final prototype. It enables applying abstract models in preliminary design stages to evaluate rough security aspects of the system. During the design process the model is refined and will be more accurate. Each refinement step provides the opportunity to perform associated security analyses of the current design phase. When software prototypes are developed it is possible to attach and assess them in combination with the simulated parts of the system. The virtual prototype of the system is beneficial even when the physical prototype is ready. Monitoring internal states of the system when the complete system is halted in physical prototype is cost and time intensive in comparison to virtual prototype.

Envisioned design flow

Figure 1 represents the proposed approach for security analysis of embedded systems during system development phases. The generic idea lays on the security-based V-model. It also inherits the iterations from iterative model for system development. Exploring overall view of the framework at a glance, comparable to V-model system design begins at very abstract level on the left side and passes through the next steps by defining more detail to the design until producing final prototype. At the right side of the flow, set of security evaluation and tests corresponding to the development level is defined. It follows a sequential design process on the vertical axe and system verification is planned in parallel with a corresponding stage of development in horizontal manner. Security specifications of the system present validation factors for the security analyses through the design process. In an overall view, after defining security requirements this framework follows several phases in a vertical axe starting from top. Each phases consists of three steps in horizontal axe which starts with design and development on the left side, performs tests and compares the results with validation factors on right side.

After specifying security requirements of the system, modeling phase is the next step. For this phase of system design, we defined an approach to assure that design decisions are consistent with security requirements. At the end of each analysis on the right side of the flow the results will be compared with the validation factors. Moving forward to the design flow happens only when the results of the current analyses pass validation factors. Otherwise the next iteration will begin from top of the flow. Each iteration starts with requirements specification, refines the model to resolve security weaknesses and as long as passes the requirements validation, proceeds through design progress.

Each iteration not only is sophisticated enough to perform corresponding analyses of the current abstraction level, but also provides information for penetration testing of the final virtual/ physical prototype. Information such as potential attack vectors as well as the must be protected assets extracted from models of the system can be considered as guidelines to steer penetration tests. Dynamic and static analyses on the architectural design on the other hand sound the alarm of hidden dependencies and potential vulnerabilities for penetration testers.

Subsequently, each part of the presented framework will be explained in this order: Security specification, Model designing and evaluation, architectural design and assessment, software analysis and penetration testing as the final goal.

At the first step system designers will define security requirements and specifications. To carry out this phase, system designers may discus with stakeholders about their prospect security features of the system. Security specifications in this phase may vary from general features to well detailed requirements. The outputs here are utilizable for security validation in next phases. Well specified security features and requirements come up with more precise specification validation for next steps.

In the next step designing phase will be started from very abstract level with modeling. The first model of the system will be sketched considering security requirements. The modules and their relations, data type and methods of the modules as well as required hardware and software for the system will be specified here. At the end the model will be augmented with security features and mechanisms. We defined a security profile which helps the designers to integrate security relates information to the model [12]. Specification validation is the next step of this phase which applies the output of modeling analyses and assesses security measure of the system based on the specification from the first phase. If the specification validation passes the expected standards, the development process can proceed through the layout. Otherwise, the next iteration will be launched.

In the next phase, designers will design a secure architecture for the system. Accomplishing this objective, an abstract model of system architecture will be simulated with the help of architectural modeling techniques such as Transaction Level Modeling (TLM). More detail will be added to the system model from previous phase and the transaction between the modules will be simulated. This phase carries on the procedure by a static/ dynamic analysis threat modeling to find out data dependencies and potential threats. Threat models will be facilitator in the next phases for the penetration testing. Based on the results of the analyses, specification validation step will check if defined security specifications are endangered. If not, the next iteration will be started and the model will be modified. Otherwise, designers can go to the next phase.

When software prototype is available, thanks to the Virtual prototyping it can be attached to the simulated parts of the system in order to perform analyses. In this way, we not only can perform penetration testing on the solid software, but also try to penetrate the software which is laid on simulated hardware. The output of the penetration testing will be compared with the security specifications and based on the results, designers decide to launch the next iteration or go to the next phase.

The final goal in this process is to integrate the software to the real hardware and perform penetration tests on the final prototype. This phase helps the developers to analyze final prototype and resolve the last security flaws before producing the product in the huge amount. Even in this phase that physical prototype is available, applying virtual prototype to

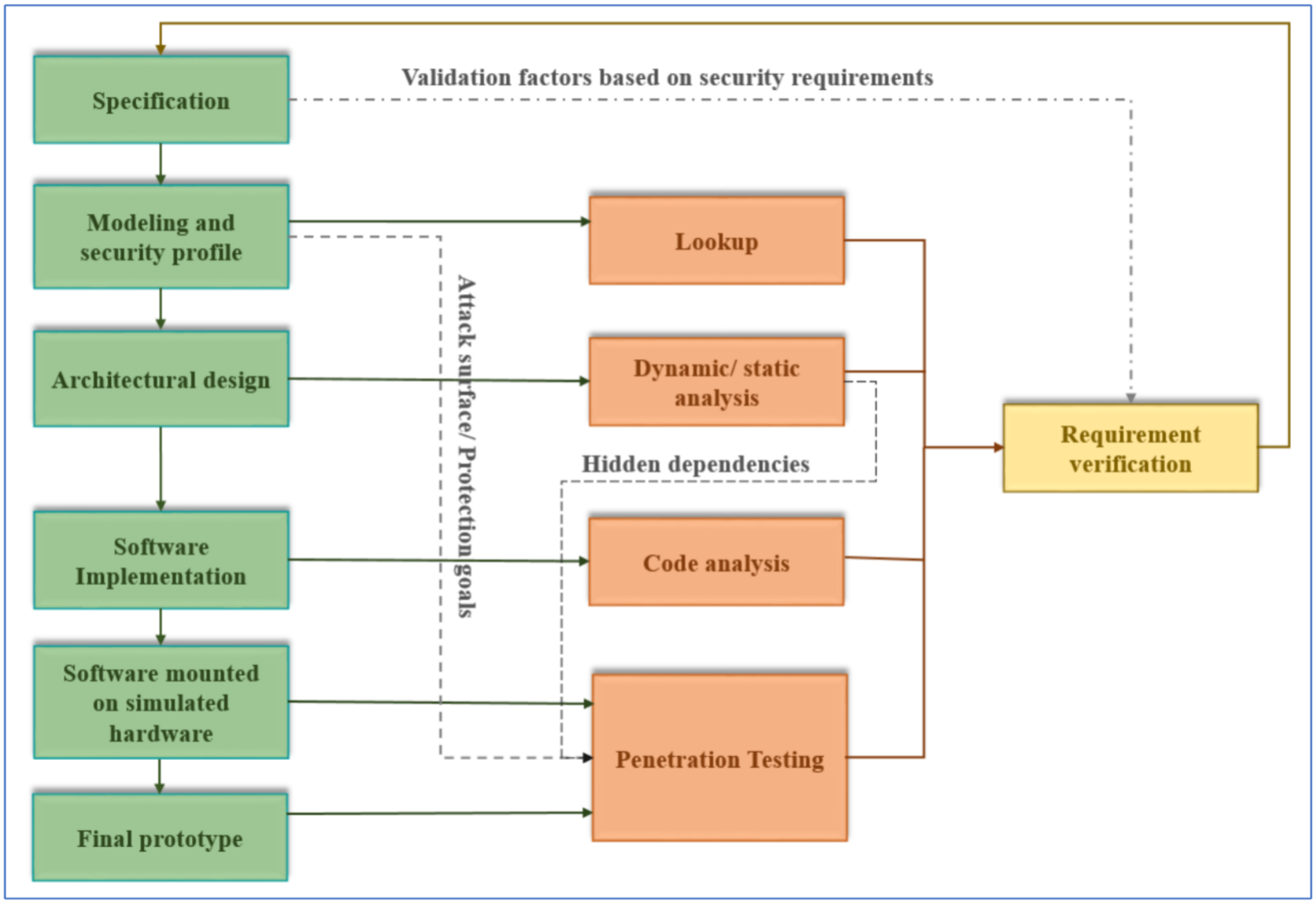

Fig. 1. Proposed approach

perform the tests in several situation has benefit over physical prototype. In the scenarios which may lead to crash the whole system or may damage parts of the system, monitoring behavior of the system on virtual environment has advantage over system evaluation in physical prototype.

V. Implementation

In order to support presented security analysis framework, several techniques are offered. As mentioned earlier, specifying security requirements plays an important role in designing secure systems. By a well defined security specification stakeholders, designers and security experts are able to properly address their desires. We propose three-step method to specify and manage security requirements from coarse system aspects to detailed technical features. As the next step in our security analysis approach, supporting modeling, a security profile based on UML (Unified Modeling Language) with the focus on connected embedded systems are offered which goes along with a look-up function to search in vulnerability databases to find vulnerable parts of the system. Architectural analysis of the system is the next step in our proposed process and in order to bring it into practice, we applied SystemC to simulate the system in TLM (Transaction-Level Modeling) level, afterward taint propagation analysis as a dynamic analysis technique is employed. As the last step we employed output results of previous parts to provide information for guided penetration testing on the final virtual and physical prototypes. In the following, the applied techniques are described in detailed.

A. Security Specification

Requirement specification is the entry point of a system development process. Following section explains the proposed approach for a consistent and guided security requirements modeling. We provide a three-step requirement modeling, which supports system designers to define security requirements from general specifications to fine-grained system architecture's security features. At very first steps of design process, security specification starts with general security aspects of the system. As the design process goes into more detail, security requirements will be shaped more clear and go to detailed categories. The requirement modeling concludes with a detailed specifications and assign security information such as protection goals (assets and data which should be protected from attackers) and attack surface (points of the system which may offer access to the system for potential attackers) to single entities in the system. The revealed information in last step is advantageous for architectural analysis as well as penetration testing. Because this last step should ensure a seamless transition into the system development process, we applied it to a traditional UML modeling approach by a developed security profile.

As mentioned, the first step in security specification brings up general questions about security issues regarding the system. The stakeholders discuss about their desire system with the system designers as well as with security experts to make a security checklist. Questions such as What are the important parts of the system which should be protected from attackers, Who are the potential attackers, Which security measures are needed and so on should be answered during security requirement analysis. The answers to these questions give an overall perspective about the security aspects of the system. However, this information is qualitative and subjective and in order to apply this knowledge to design and development process, well-defined classifications and security metrics are needed.

In the second step of security specification, protection goals and potential attack surfaces of the system will be derived and categorized in the three classes of the CIA triad: Confidentiality, Integrity, Availability. To illustrate, confidentiality for assets refers to system properties which should be protected in order to assure their confidentiality. On the other hand, confidentiality in attack surface discloses parts of the system which are potential attack point to endanger confidentiality of the system. From this general specification, all relevant information will be extracted in the next step.

The last step maps the information from the second layer about security requirements to a well-defined schema. The goal is to map detailed security information to system architecture. Accordingly, several fine-grained categories as well as parameters are defined which fulfil system architecture with well-defined information about assets and potential attack vectors. Figure 2 depicts proposed security requirement method in three levels.

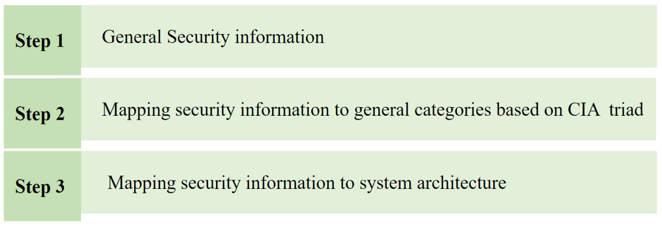
Step 1
General Security information
Step 2
Mapping security information to general categories based on CIA triad
Step 3
Mapping security information to system architecture

Fig. 2. Three steps of proposed security requirement analysis model

B. *Security specification modeling*

Presented model-based specification describes the system to be analyzed structurally, safety concepts and security mechanisms, attack surface, as well as protection targets. The modeling is also used for documentation of the analyses. The specified information is automatically integrated into the analysis to reduce the manual effort for the user. In this section, we highlight a modeling approach used to guide the security analysis with VPs. We use UML 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.

As Figure 3 shows, the proposed model-based security analysis approach consists of two components: Graphical modeling environment, which is composed of UML models augmented with security features as a security profile and a lookup function which can be connected to a vulnerability database. This approach enables the designer to refine security requirements in the same development environment as the system design. Our proposed security modeling approach provides models for security requirements of the system and helps to discover the inherent weak points of the system architecture or chosen implementations by specifying the protection goals of the system, potential attack points as well as additional security related documentation.

Our presented security profile offers three categories of stereotypes and tags to add security-related information to the model. The first category consists of five stereotypes and related tags which help the designers to mention protection goals on the model. Second category suggests five stereotypes and their related tags to add information about potential attack vectors to the model. Finally, with the aim of offering documentation-based information such as indicated security mechanisms and software version our last category of stereotypes is suggested. The proposed security profile can be attached to the UML models to entitle the information about protection goals and assets, weak points and attack surfaces, as well as documentation-based information. These models can later on be beneficial for validation, e.g., with static analysis and vulnerability assessment. In addition, the model simplifies manual analysis, such as penetration testing, by boosting the documentation and helps identifying underlying potential design flaws, e.g., by enabling an automatic lookup in well known security vulnerability databases.

Drawing the advantage of documentation-based information form presented security profile, our model-based security analysis approach offers connection to vulnerability data bases which provide latest discovered security vulnerabilities for embedded systems. We presented this function in our lookup function. Vulnerability database describes vulnerabilities, affected assets, and the solutions to mitigate the issue. There are several vulnerability databases such as as the ISS (Internet Security Systems) X-Force database [13], Symantec / SecurityFocus BID (Bugtraq ID) database, and the Open Source Vulnerability Database (OSVDB) [14] which aggregated a broad range of publicly disclosed vulnerabilities, including Common Vulnerabilities and Exposures (CVE) [15]. CVE, run by MITRE collects vulnerabilities and puts them in a standard format. for the sake of connection to up-to-date vulnerability databases, the influence of design decisions on the actual system will be evaluated in very early stages.

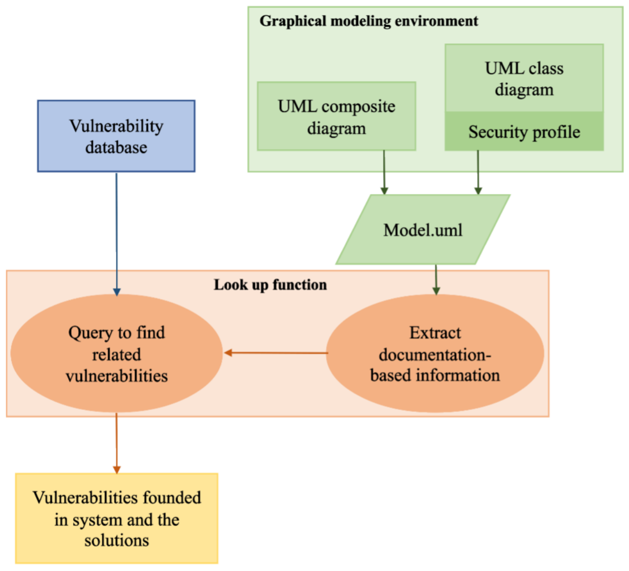
Graphical modeling environment
UML class diagram
UML composite diagram
Security profile
Vulnerability database
Model.uml
Look up function
Query to find related vulnerabilities
Extract documentation-based information
Vulnerabilities founded in system and the solutions

Fig. 3. Proposed modeling environment

Clarifying lookup function, security-related information will be extracted from the security profile and the function searches for probable vulnerabilities in vulnerability database based on this information. Information such handshaking protocols, applied cryptography methods, Encryption protocols and software version enables finding potential vulnerabilities based on published vulnerabilities and exposures. The designers will be able to find out vulnerable parts of the system at modeling phase and to rectify the model to sketch secure layout for the system.

C. System Architectural Analysis

System architecture is an important aspect in the design of trustworthy systems. We can develop security level by means of a suitable architecture and thus logical structuring, even in early development phases. The proposed approach offers system architecture assessment in design phases and as a result, prevents architectural weaknesses. However, in embedded systems design and development, hardware is only available at the latest phase.

With the help of abstract virtual prototypes, which represent the architecture of the system to be developed and target-platform-specific IP components, analyses for architectural security issues are to be carried out. Due to the fact that the virtual prototypes describe the hardware / software structure in a software-based model, analyses which have traditionally been applied to the pure software evaluation can now be applied to the virtual total systems (hardware / software). There are methods such as Dynamic Taint Propagation or Symbolic Evaluation. By applying these methods on VPs, analyses that are currently used for input validation and filtering in software implementations are used for hardware architecture analysis e.g. to inspect gateways or controllers.

The aim of this section is to verify the effectiveness of security models such as access rules, input validation, or filtering of potentially counterfeit messages. This means that correctly functioning security mechanisms are adopted and modeled and their correct use verified in this type of analysis. From the point of view of system architecture, for example, the effectiveness of compartmentalization, the separation of critical and uncritical system parts, is to be checked. Several techniques such as static and dynamic analysis in this step facilitate architectural assessment of the system. In static analysis methodology the source code will be assessed at rest in order to detect security flaws such as input validation, numerical errors, path traversals, and race conditions. Dynamic analysis procedure on the other hand assesses the application during run time with the focus of detecting conditions during which the application can be exploited. Various tools and techniques are offered in these areas which let security experts find out security condition of the software. As a capable method, we offer taint propagation analysis [16] to perform dynamic analysis on the simulated system.

1) Simulation approach in virtual prototype framework: The presented approach uses a subset of the UML to specify VP and its security analysis related information. In this section, the link between modeling and VP should be described. An overview is given in Figure 4. The simulated model is generated from this specification. The simulation contains all system components required for one simulation run. This simulated model 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 model-based VP specification. Different security analyses, such as dynamic data flow analysis or manual penetration testing are based on the executable simulation.

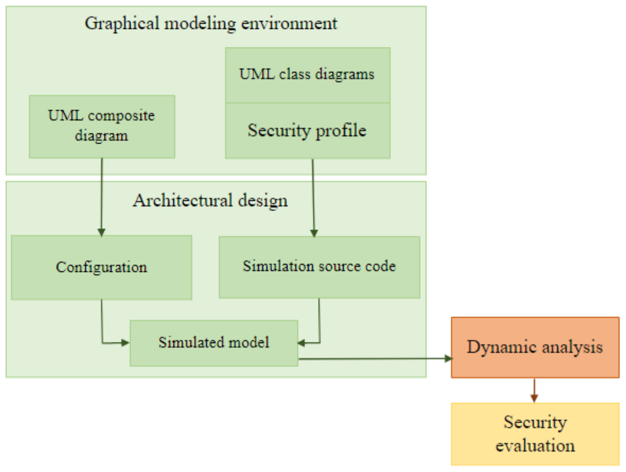

Fig. 4. Modeling framework for the VP-based security analysis

The presented approach uses event-driven SystemC [17] simulation language, a standardized modeling language with a lot of EDA (Electronic Design Automation) vendor tools support. SystemC covers different abstraction levels of hardware and software systems and enables both the modeling of software applications as well as digital/analog electronic components. SystemC provides abstract 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. The overall goal is to execute security analyses combined with risk assessment.

D. Penetration Testing

By the time that final software is available, testing pure software will be started. Several software security analysis techniques with the purpose of vulnerability detection may be performed in this phase. An appropriate software test is structured around elements such as the risks, assets, potential threats and vulnerabilities. Our evaluation framework facilitates software analysis by providing needed information extracted from previous steps.

However, the advantages of presented framework are not limited to pure software testing; Instead, the framework proposes significant gain by offering security analysis for the software in combination with simulated hardware. Integrating already developed software into the virtual prototype offers an overall view of the system in a way that general penetration testing will be possible.

In order to perform a successful penetration testing the following six phases are critical: Recognition, Scanning, Gaining access, Maintaining access, Exploitation, Clean up and Reporting

Security analyses performed in previous stages of presented evaluation framework, in addition to major benefits in reaching efficient design decisions in early stages, propose information to guide final penetration testing. The rest of this section is devoted to describe penetration testing phases separately and to represent how proposed approach provide information to these steps.

1) Recognition: In recognition phase penetration testers try to get as much information about the target as possible. Information such as the scope of the system, network and interconnection architecture as well as the software and hardware features should be extracted.

Our proposed approach offers comprehensive and well-defined information to penetration testers. Proposed security requirement specification sidelong system requirement specification gives general information about the system, potential attackers, data and assets which should be protected and desired security features. Our security profile on the other hand with its documentation-based information offers knowledge about the software, encryption algorithms as well as other security mechanisms such as authentication and authorization methods.

2) Scanning: In scanning step, testers try to identify vulnerabilities of the target. Scanning the network with scanning tools, identification of open share drives and running services are the measures which will take place in this phase. static and dynamic analysis alongside vulnerability scanners are powerful techniques to detect vulnerabilities.

Presented security requirement specification, modeling approach and architectural analysis offer information as guideline to find vulnerabilities. Attack surfaces give testers the first impression about weaknesses as entry point to penetrate the system. On the other hand, protection goals mentioned in the model, guide testers to concentrate on the assets with essential data or the services which should be available all the time as the target of attack. Lookup function reveals known vulnerabilities for chosen software or protocols. By detecting any know vulnerability in modeling phase designers modify the model in early stages before penetration testing; However the information extracted from know vulnerabilities give direction to tester to find more vulnerabilities. Further considerations for the testers derive from architectural analysis to sound the alarm about hidden dependencies and possible track between attack point and the targets.

3) Gaining access: Once the weaknesses are identified, the next step is gaining access to the system by exploiting vulnerabilities. It should be mentioned that not all of detected vulnerabilities are exploitable and the ones which are vulnerable enough to provide the access into the target should be identified. Integrating already available software with simulated hardware will be the first detailed development of the system which proposes the chance to penetrate the prototype. In this step the entry points to the system, authentication and authorization mechanisms and the security protocols are developed in the software. The network, the hardware as the environment to mount the software and all the connections is simulated now. By performing tests on this virtual prototype weaknesses of the system as entry points for unauthorized user will be disclosed.

4) Maintaining access: Maintaining access as the next step of penetration testing is possible on the final virtual prototype by rebooting, resetting or modifying the system to find out if the access will be maintained under different conditions.

5) Exploitation: Exploitation phase is the point that actual damage to the system will be brought up. When the penetration testers attain access to inside of the compromised system, they try to promote their access privilege within the environment, subsequently performing any number of additional actions. With the administrative privilege the testers are able to find security weaknesses in other areas and resources, such as weak connections, unguarded access to critical data, implementation's weakness in gateway and control devices which interfere on several buses and inner network or ineffective account or password management. Final virtual prototype presents an environment for the penetration tester to exploit the system without concerning about the potential expensive damage to the system. Simulating time behavior of virtual prototype helps to evaluate the system under timing-related attacks such as denial of services. By integrating product-related software prototypes, implementation errors or weak points, such as incorrect, insufficient checking of value ranges, use of copy operations without validated, fixed upper limits can be checked. Further implementation aspects such as the use or generation of random values, e.g. in the case of key streams in stream ciphers, can be analyzed. Sometimes, the mechanism used is fundamentally correct, but implementation weaknesses, such as reuse of generated random values, reduce the effectiveness of applied mechanisms. Other aspects of the system security model can only be verified with the presence of the partially manually optimized binary code. These include, for example, buffer overflows with effects on the stack structure and change of return addresses. Virtual prototype as the latest step of this approach assists security analysts to evaluate these features.

6) Clean up and Reporting: Once the penetration testing is completed, the tester will find a way to clear any evidence and trace of compromising the victim system including event logs in order to remain anonymous. Eventually, as the final goal of the penetration testing, a detailed report will be written. Detailed information about vulnerabilities, sensitive data accessed during the test, complexity and the amount of the time to exploit an asset and the way to delete the evidences should be reported in this step. Well-defined information in a standard manner which will be comprehensible to designers is beneficial to refine the design. With the help of proposed three steps for security specifications, security profile in modeling environment as well as offered solutions for known vulnerabilities this intention will be achievable.

VI. Conclusion

In this paper, we applied virtual prototyping to present a security by design approach with the intention of security assessment in each design and development phase. This approach is based on two SSDLC models: V-shaped model and iterative model. The proposed framework gains benefits from both V-model and iterative model of SDLC, however provides advantages over applying each of the models separately. Design and development process takes place in sequential manner from very abstract level to detailed developed product. It also derives a benefit of verification and validation perspective of V-model by providing a associated testing phase for each corresponding development stage. Proposed framework inherits the iterations from iterative model. However unlike the iterative model, in each iteration as well as the requirements are satisfied development process goes further into design flow and more detail will be added.

Yet, the strength of the presented approach is based on employing virtual prototyping to fill the gaps between already designed and developed system modules at each step and final prototype. Virtual prototyping supports simulation in different abstraction levels; moreover, it offers detailed simulation as the final prototype. These features in combination with SDLC models enable system designers to evaluate design decisions at very early stages, to analyze system architecture from security point of view, to perform penetration testing way before that physical prototype is ready and as a result present a cost and time efficient approach. This approach has profits even when the final physical prototype is ready. Sometimes, performing tests on physical prototype may damage parts of or even the entire system. In these situations final virtual prototype could be a practical alternative to prevent loss during the tests.

Acknowledgement

This research is partially funded by the German Federal Ministry of Education and Research (BMBF) within the project COMPACT (grant number 01|S17028C) and project Scale4Edge (grant number 16ME0126).

References

[1] L. A. Bygrave, "Security by design: Aspirations and realities in a regulatory context," *Oslo Law Review*, no. 3, pp. 126–177, 2022.
[2] M. A. Rather and M. V. Bhatnagar, "A comparative study of software development life cycle models," *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 4, no. 10, pp. 23–29, 2015.
[3] K. Pohl, M. Broy, H. Daembkes, and H. Hönninger, "Advanced model-based engineering of embedded systems," in *Advanced Model-Based Engineering of Embedded Systems*. Springer, 2016, pp. 3–9.
[4] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: A systematic mapping study," *Computer Standards & Interfaces*, vol. 50, pp. 107–115, 2017.
[5] M. U. A. Khan and M. Zulkernine, "Quantifying security in secure software development phases," in *2008 32nd Annual IEEE International Computer Software and Applications Conference*. IEEE, 2008, pp. 955–960.
[6] S. Tverdyshev, "Security by design: Introduction to mils." in *MILS*, 2017.
[7] F. Köster, M. Klaas, H. Q. Nguyen, W. Brenner, M. Brändle, and S. Obermeier, "Collaborative security assessments in embedded systems development," *NSTICC Pr.*, 2009.
[8] J. Geismann, C. Gerking, and E. Bodden, "Towards ensuring security by design in cyber-physical systems engineering processes," in *Proceedings of the 2018 international conference on software and system process*, 2018, pp. 123–127.
[9] A. Ferrante, J. Milosevic, and M. Janjušević, "A security-enhanced design methodology for embedded systems," in *2013 International Conference on Security and Cryptography (SECRYPT)*. IEEE, 2013, pp. 1–12.
[10] E. Mougoue, "What is the secure software development life cycle (sdlc)?— synopsys," 2016. [Online]. Available: https://www.synopsys.com/blogs/software-security/secure-sdlc/
[11] M. E. Whitman and H. J. Mattord, "Principles of information security, course technology," *Google Scholar Google Scholar Digital Library*, 2012.
[12] Y. Mahmoodi, S. Reiter, A. Viehl, O. Bringmann, and W. Rosenstiel, "Model-guided security analysis of interconnected embedded systems." in *MODELSWARD*, 2018, pp. 602–609.
[13] Ibm internet security systems. x-force. [Online]. Available: https://www.ibm.com/x-force
[14] Open source vulnerability database. [Online]. Available: http://www.osvdb.org
[15] Cve list. [Online]. Available: http://cve.mitre.org/
[16] M. G. Kang, S. McCamant, P. Poosankam, and D. Song, "Dta++: dynamic taint analysis with targeted control-flow propagation." in *NDSS*, 2011.
[17] I. S. Association *et al.*, "Standard systemc language reference manual," *IEEE Std*, pp. 1666–2011, 2011.