Model-based Security Engineering for Secure Systems Development

An approach for Software Engineering

Armin Lunkeit
OpenLimit SignCubes GmbH
Berlin, Germany
armin.lunkeit@openlimit.com

Hartmut Pohl
softScheck GmbH
Sankt Augustin, Germany
hartmut.pohl@softscheck.de

***Abstract*— Security of software and systems is becoming more and more important in the context of the rapid rise of distributed communication systems and their use into the private life of each individual. The development of new software is also accompanied by an immense time and cost pressure. The aspects of IT-security are therefore often not considered within the software development process. The aim of this work is to contribute to the improvement of the integration of security engineering into software engineering. A model-based approach for determination of assets, security objectives, threats, and attacks is presented. The derivation of functional requirements for the software development process from these artefacts is explained.**

Keywords security engineering; software development; model-based engineering

I. INTRODUCTION

Secure systems development requires a profound knowledge of software engineering and security engineering. Both disciplines overlap and the integration and interaction between them is important throughout the software lifecycle. This includes requirements engineering, architecture and design, implementation, test, operation and maintenance. Nevertheless, security engineering is not yet established as an integral part of standard software development processes.

The motivation of the authors for dealing with this topic results from the practical experience in software and system development. Among other things, the development and Common Criteria Evaluation of a Smart Meter Gateway (SMGW) was accompanied from the early beginning. Moreover, the design and development of other, comparable devices in the eHealth sector as well as set-up machines was supported. In all these developments it became apparent that a) basic techniques of security engineering and their consistent use in the software development process are not well known by software engineers and b) frameworks like the Common Criteria have a level of abstraction that is too high to be used easily in the development process.

In accordance with this experience, a number of papers identify the lack of integration of security engineering into software engineering as one of the major challenges. The lack of consideration of security requirements in the development phase leads in consequence to inadequate security measures, the late recognition of architectural and security problems and concomitantly to economic implications, which result from the late remedy of these problems [1], [2].

For this reason, the early identification and integration of potential and existing security requirements by security requirements engineering is an important part of the requirements phase. The work presented in [2], [3], [4], [5], [6], [7] addresses this issue and has led to different approaches. The use of these approaches leads to different results with regard to the resulting security requirements [8]. This is due to the different characterizations and methods used to determine security requirements [9], [10], [11], [12], [13]. The characterization of the security problem and the context of the identified security requirements carried out by security engineering therefore have a significant influence on the aforementioned aspects of software development and justify the necessity of integrating security engineering into software engineering.

This paper is structured as follows: In the first part, an overview of approaches in security engineering is given. This is not intended to be a comprehensive list of all possible variants and is limited to the determination of security requirements. Aspects of implementation and testing or operation are excluded. In the second part, we introduce the approach we use. This was developed in connection with a PhD thesis [44]. In the third part the use of the security engineering approach in the development of a smart meter gateway (SMGW) is shown. Finally, an outlook on future work is given.

II. STATE OF THE ART

A. Model driven software engineering

In the requirements and architecture phase, various model-based approaches are used. From the object-oriented methods, the *Unified Modeling Language (UML)* and the goal-oriented approaches *i** and *Tropos* were selected. i* is the basis for different requirements phase approaches, e.g. Tropos and its

evolution *Secure Tropos*.

The Unified Modeling Language (UML) is a modelling language maintained by the Object Management Group (OMG). The core concept of UML is the modelling of static structures and dynamic behaviour. Static structures are represented using classes and discrete objects, and relationships are expressed with inheritance and mapping. Classes can map static structures of arbitrary states and, like objects, have attributes. Dynamic behavioural descriptions are mapped with states and their transitions, sequences, and actions.

With use cases and their diagrams, UML offers a method for requirements analysis. Use cases can be linked through extensions and inclusion. With this approach, functional requirements can be modelled, while this is only indirectly possible for non-functional requirements. Various extensions of the UML have been proposed to integrate the mapping of non-functional requirements into the UML [14], [15], [16], [17]. In the domain-specific context, the possibility of profiling is used. The two approaches *UMLsec* and *SecureUML* extend the UML with aspects of IT security.

i* is a goal-oriented framework for requirements determination and analysis and was described in 1995 in the dissertation by Eric Yu ([18]). Originally i* was designed to model business processes related to software development. A central idea in the i* framework is the interdependence of actors to achieve a strategic goal, to provide a resource, or to perform a task. In the model of the strategic principles, the organizational context and the internal relationships of the components are depicted. A central concept in i *, which is also used by the further development Tropos and Secure Tropos, is the distinction in the definition of the goals. *Hard-Goals* are strategic goals whose achievement is essential. *Soft-Goals* model goals whose fulfilment conditions are not clearly defined. This concept is transferable to functional and non-functional requirements. The further development of the approach takes place in the Tropos framework [19].

Tropos [20] is a modelling approach that takes advantage of the agent-oriented paradigm (AOP). While i* was originally developed to better model processes, Tropos is a framework for gathering and modelling requirements right up to the system architecture. The core concept of Tropos is the development of a model of the target system and its environment, which is gradually refined in an iterative process. The methodology supports various software development activities [20]. Unlike UML, Tropos does not provide a mapping to the implementation. The usability of Tropos in software engineering is therefore primarily located in the requirements phase and the early architecture and design phase [20], [21], [22], [23], [24].

B. Security engineering - analysis and design techniques

The *CORAS* method is a risk-oriented approach that can be used to investigate technical and non-technical issues. Eight steps are defined from the preparation for the analysis to the identification to the treatment of existing risks. Acceptable risks are not pursued in the analysis, while unacceptable risks must be dealt with. A central part of the method is the graphical analysis. This reduces the complexity of the examination and documents the causes of a potential risk in a comprehensible manner. There are various types of charts defined (asset, threat, risk and treatment charts) that help in risk analysis. The CORAS language for describing, documenting, and analysing threats and risks was originally defined as an UML profile and has evolved into a domain-specific language (DSL). The original UML profile has become part of OMG's profile "*UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification*" [25].

The *fault tree analysis (FTA)* is used to determine the reliability of a system and enables the estimation of the probability of failure [26]. It is used in safety engineering and is important in the field of avionics, military technology and probabilistic safety analysis in the operation of nuclear power plants [27]. In security engineering, fault tree analysis is used in risk-based approaches such as CORAS [28].

Fault trees use AND and OR expressions to link individual events and merge them at the root into an undesired top event. It uses a negative logic; a value of 1 indicates that the error event has occurred. In order to determine the probability of failure and thus the probability of occurrence of the undesired result the probabilities of occurrence of each individual negative precondition and the AND and OR expressions are evaluated. The procedure of the fault tree analysis is related to the attack trees (attack trees) [29]. The method is used in procedures such as CORAS and Threat Modelling and in combination with other approaches of security engineering.

Threat Modelling is a technique for detecting threats directed against a system. The analysis takes place in tree structures and shows the preconditions leading to a successful attack. These preconditions, like a fault tree, are linked using AND and OR statements. The calculation of the probability of occurrence differs from the fault tree analysis, since the associated probabilities are in the context of the assumed attacker and his capabilities.

The prerequisite for the application of threat modelling is the knowledge of the values to be protected, the attack surface and the characterization of the attacker. The analysis is done using dataflow diagrams, which simulate threats and attacks. Elements of threat modelling are integrated in various methods and procedures, for example as part of the Security Development Lifecycle (SDL) [30], [31], in the CORAS methodology or in the Common Criteria. The results of the threat modelling can be used as a basis for the determination of security requirements (see [32]). However, this is only possible if the security objectives for the system and its assets have been precisely determined beforehand. If this is not done, the image of the threat situation is discarded and can generate inconsistent security requirements. The identification of attack trees and the use of data flow diagrams to identify potential

attack paths as well as classification and risk assessment require comprehensive knowledge of the planned or actual implementation of the system.

Secure Tropos is an extension of the Tropos framework and can be used in the requirements and architecture phase [33] as well as in the test phase [34], [35]. A graphical notation is used to model design goals of a system.

UMLsec is a UML profile for modelling aspects of IT security [36], [15]. The profile supports the phase of design and architecture, from the capture of use cases to sequence and state diagrams. In [15], chapter 6, both the connection between model and code as well as the extraction of test cases from the existing model are discussed, but not deepened further; the focus remains on design and architecture. New stereotypes, boundaries (tags) and constraints are introduced to model and validate security requirements.

SecureUML is a specialized extension of the UML meta model with additional stereotypes for role-based access permissions (RBAC) [37], [38], [39], [40], [41]. Modelling of attacks and secure protocols is not supported.

The Common Criteria are a framework for modelling and evaluating the IT security of products. Functional security requirements offer a technology-neutral abstraction and leave the concrete implementation of the security requirements to the developer of the product. The evaluation framework is asset-centric. Based on the values to be protected, countermeasures are defined to minimize threats and prevent successful attacks. It is crucial to determine whether the measures implemented are adequate and treat the security problem correctly. The modelling of the security requirements is based on functional security requirements. These are used in the protection profile and in the security specification for modelling the security objectives. The Common Criteria offer in [42] a catalogue of existing components for just that purpose. The Common Criteria define a methodology for modelling security requirements, but do not make any necessary contribution to the design. The main focus is the evaluation of IT security. The Common Criteria provide a snapshot of the IT security of a product and are methodically poorly prepared for changes in the product or operating environment.

III. MODEL BASED SECURITY ENGINEERING

This section introduces our approach to security engineering. It provides the actors in software engineering with a means to model the security problem to be solved. This addresses one of the core problems in software engineering - the evaluation of the suitability of selected security mechanisms. We define the key terms and then introduce a meta-model that focuses on the relationships between assets, security objectives, threats, attacks, and derived functional security requirements. We call this *Security Problem Definition*. On this basis, a security problem analysis approach is presented, the application of which will enable the effectiveness of the chosen security mechanisms to be assessed in the context of existing threats and attacks.

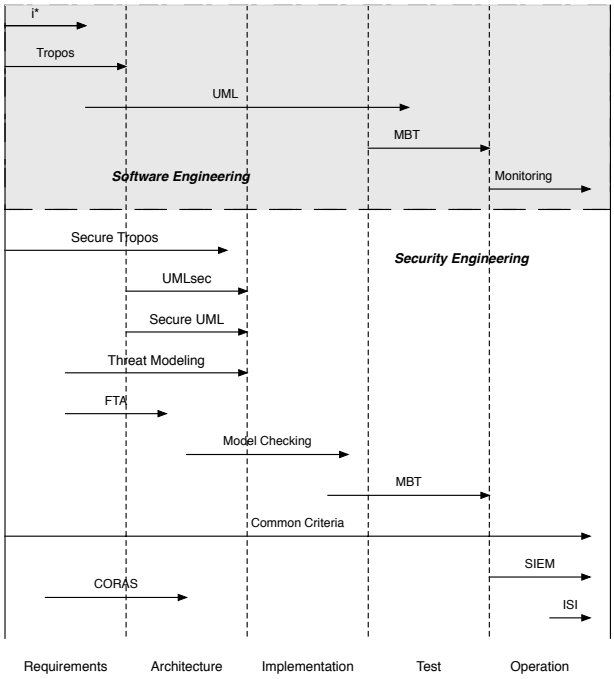

Fig. 1. Methods in context of software lifecycle

A. Perspectives

The model of the security problem identifies and characterizes assets, applicable security objectives, threats, attacks and attackers, and their respective potentials. The security problem introduces two perspectives.

1) The *socio-technical perspective* models the assets to be protected, applicable security objectives and threats from the point of view of external actors interacting with the system. The system is considered a black box.
2) The *technical perspective* models the assets to be protected, applicable security objectives and threats from the system viewpoint. Internal assets are identified, their security objectives and applicable threats are characterized and linked to the objectives of external actors.

Both perspectives are stakeholder perspectives. The socio-technical perspective models the view of entities, which have an interest in the implementation of the requirements. It examines the interaction of external actors with the technical system and describes values, security objectives and threats. External actors identify other assets and security objectives as with an exclusively technical view of the system. The technical perspective deals with the technical aspects of the system and provides the mapping to the system design and implementation. In this case, the project participants or the manufacturer are to be considered as stakeholders. The use of the model of the technical perspective ensures the fulfilment

of the security goals identified in the socio-technical perspective.

Using both perspectives provides the advantage of a comprehensive analysis of all aspects of the security problem. The technical perspective takes up the goals of the external actors and the roles they have implemented. It focuses on those values and security objectives that are prerequisites for achieving the objectives of external actors. These values and security objectives and the corresponding threats are irrelevant to the black-box view of the socio-technical perspective of the external actor.

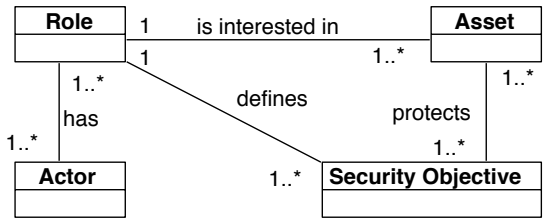

Fig. 2. Relation between Actor, Role, Asset and Security Objective

An actor can take on several roles, as well as a role of several actors can be perceived. The role has an interest in a value to be protected (asset). The role also defines the security objective for the value to be protected. The security objective protects the asset. These relationships apply equally to both perspectives. Security objectives are defined by, but not limited to, the terms authenticity, integrity, confidentiality, and availability. The schema allows the definition of further security objectives, which are then described in textual form.

B. Scenarios

A cross-perspectival concept is the scenario. Scenarios are used to outline the context of the assets. Not every scenario involves all the assets to be protected. Likewise, the security goals may vary for a value to be protected between scenarios.

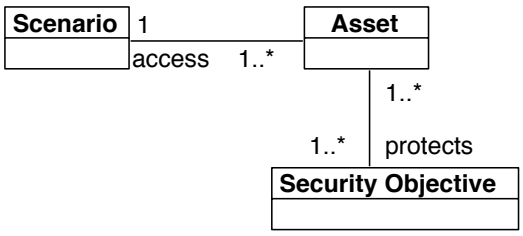

Fig. 3. Scenario concept

C. Threats and attacks

Key elements for modelling the security problem are threats and attacks. Threats subsume possible attacks directed against an asset. Threats are described more abstractly and do not deal with the technical implementation. The initial description of the security problem uses this level of abstraction. In addition, the concept of the security mechanism is introduced. Security mechanisms are used individually or in combination to protect an asset against threats and their associated attacks. This distinguishes our model approach from such approaches as the Common Criteria. It works with the abstract concept of threat, while this approach takes into account specific attacks. This distinction is important because threats can be implemented in a variety of ways, and addressing generic threats does not always reveal all the resulting security requirements.

D. Meta model of the security problem definition

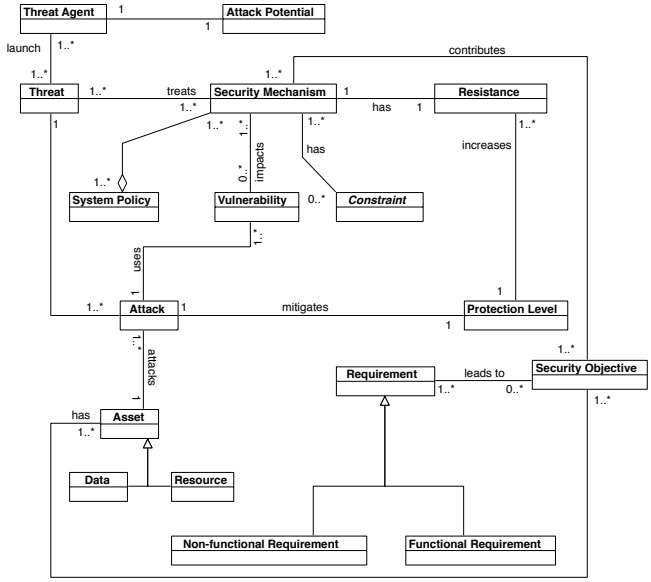

Fig. 4. Meta model of the security problem definition

The security problem model places requirements, security objectives, assets, and security mechanisms in a context of threats, attacks, and vulnerabilities. In addition to determining the relationships between these elements, the definition of potentials is a key element of the approach. In order to enforce the security objectives of an asset, the level of protection obtained from the security mechanisms must be equal to or greater than the potential of the attack. To illustrate these relationships, the model elements attack potential, resistance and protection level are introduced. With this foundation, the security problem models the relationship to the requirements that are raised to solve it.

The central element of the model is the asset. Assets may be resources or data. Assets have associated security objectives. Security objectives lead to requirements that may be functional or non-functional. This approach establishes the link between assets, security objectives and requirements. Security objectives are always described succinctly in terms of integrity, confidentiality, authenticity. This distinguishes the approach from the Common Criteria approach, which formulates broader security goals. This more comprehensive representation is roughly comparable to security mechanisms in our approach.

New elements are *attack potential*, *resistance* and *protection level*. Attack potential qualitatively characterizes the abilities of an attacker. The attacker is in connection with

one or more threats that are implemented by one or more attacks. The implementation of an attack corresponds to the capabilities of the attacker characterized in the attack potential. This characterization is a significant change to abstracted threats, as the abilities granted to the attacker are included in the analysis. The resistance characterizes the effectiveness of a security mechanism. Both are in a 1: 1 ratio, with the security mechanism countering a threat. Resistance contributes to the level of protection which counteracts an attack in the context of the assigned attack potential and protects a value. The security mechanism contributes to the achievement of a security objective, placing security mechanisms and security objectives in context.

E. Characterization of threats

Threats subsume a number of attacks targeted against the same asset and its security objectives. Security objectives are described in the terminology of confidentiality, integrity, authenticity and should be enforced for values to be protected. Threats are directed against the protected values. The goal is the violation of security objectives.

A threat is implemented by one or more attacks. This relationship is fundamental to the characterization of threats. The term threat describes an abstraction, while the attack is the implementation of the threat characterized by technical and / or social interaction. The ability to implement one or more attacks to implement an attack depends on several factors. First, there must be an exploitable vulnerability at all. The exploitation of this attack requires several abilities of the attacker, which build the foundation of his attack potential.

F. Characterization of the attacker

An attacker is characterized by the attributes preparation time, analysis time, knowledge, expertise, specialized equipment, and social capabilities. Similar factors are used in [43], Section B.4.2.2; the social abilities are not taken into account there. However, this factor is needed to characterize attacks from the context of social engineering. Attack potential of the attacker is the result of the characterization.

TABLE I. ATTACKER CHARACTERISATION

Factor	Description
Preparation time	The entire time the attacker has for attack preparation
Analysis time	The time available to an attacker to access the system and prepare for an attack
Specialized equipment	The availability of specialized equipment to identify a vulnerability and develop an attack against a security mechanism
Knowledge	Describes what information about the design of the attacked system the attacker has.
Expertise	Characterizes the attacker's expertise to identify a vulnerability and develop an exploit
Social capabilities	Describes the ability of an attacker to manipulate people and their social environment

G. Characterization of resistance and protection level

Security mechanisms defend assets against attacks. It is estimated whether the chosen mechanism sufficiently protects the value to be protected against the respective attacks. This is modelled using resilience. As a result of the characterization of the resistance, a value between 0 and 10 is determined. A characterization that is comparable to the attacker is determined and determines what effort must be made for the attack to be successful.

TABLE II. ASSIGNMENT OF RESISTANCE

Value	Meaning	Context for characterization
0	The security mechanism is ineffective.	The attacker has stronger capabilities in all respects
1	The security mechanism is weak and has only a small contribution to ward off the investigated attack.	The attacker is inferior to the security mechanism in at least one property.
5	The security mechanism does not provide complete protection against the attack being investigated.	The attacker is inferior to the security mechanism in at least three properties.
10	The security mechanism mitigates the attack completely.	The attacker is inferior in all its properties to the security mechanism.

The mitigation of an attack can be done by combining several security mechanisms. In order to take this into account, the element *protection level* summarizes the individual contributions of the individual security mechanisms. In the model approach used, the individual contributions are added up. This approach seems naive in the first step. In fact, this approach quickly shows whether the combination of different security mechanisms is suitable for warding off a specific attack. It is indisputable that the evaluation of the interaction of several security mechanisms requires the expertise of the architect. However, it can be assumed that expert systems and AI approaches in particular may be suitable for supporting this process.

H. Graphical Representation

To simplify the analysis task and the exchange of information, a graphical representation can be used. The elements Scenario, Asset, Security Objective, Security Mechanism and Threat are defined in order to represent the relationships between these elements in a simple graphical representation (see Fig. 5). This approach addresses entirely different contexts, such as threat modelling or CORAS.

A. Evaluation of the security model

Based on the previous steps, analysis and evaluation of the security problem are performed. It is determined whether the selected security mechanisms adequately protect the identified assets against attacks. If the analysis shows that all threats and attacks are treated in an adequate manner, the derived requirements flow into the software engineering. If the

security mechanisms are inadequate, it must be decided whether the threats and attacks are considered acceptable risks or whether changes to security mechanisms are required.

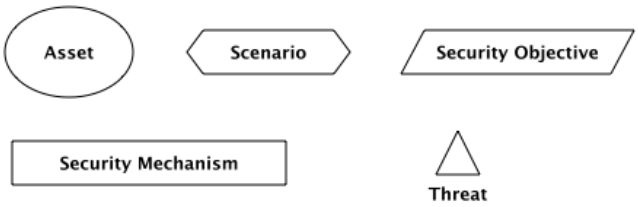

Fig. 5. Elements of the graphical representation

These decisions are made on the basis of a quantitative assessment. The quantitative assessment of the respective elements - especially the security mechanisms and the effectiveness of their combination - is not trivial. The basics of evaluating security mechanisms are publications such as technical standards and guidelines. These include, for example, the technical guidelines published by the Federal Office for Information Security in Germany, NIST standards, which provide recommendations and assessments of the use of cryptographic mechanisms as a sub-aspect. Such recommendations are not available for all conceivable security mechanisms, so a rating must be created through research. The model considers the quantitative assessment using the elements Resistance and Protection Level. The first element individually describes for a security mechanism its capabilities for mitigation of an attack. The level of protection subsumes the resilience of multiple security mechanisms, as the combination of several weaker security mechanisms may be able to mitigate a threat.

In addition to the security mechanisms, the individual attacks are to be evaluated. The characterization of the attacker shows the maximum potential of an attack. All attacks within this range are feasible for such an attacker. The model approach allows the definition of multiple attackers with different attack potentials. The analysis is based on the unacceptable attacker with the highest potential, if the characterization of the attacker shows no differences in the possibilities granted. A distinction is the accessibility. This means whether the attacker has physical or only remote access. Attacks that are available to an attacker with physical access cannot be performed by a remote attacker. With the same characterization, the analysis uses the attacker with physical access because its arsenal of exploitable attacks is greater.

B. Integration into software engineering

In order to derive functional security requirements for software engineering, the information obtained in the socio-technical and technical perspective is used. This requirement derivation transforms the security mechanisms defined by security engineering into a description that can be used for software engineering, which leads to an architecture and implementation. Since the security mechanisms counter threats that in turn subsume concrete attacks, measures against known attacks flow into the formulation of the security requirements. This prevents vulnerabilities that can be used as a basis for successful attacks.

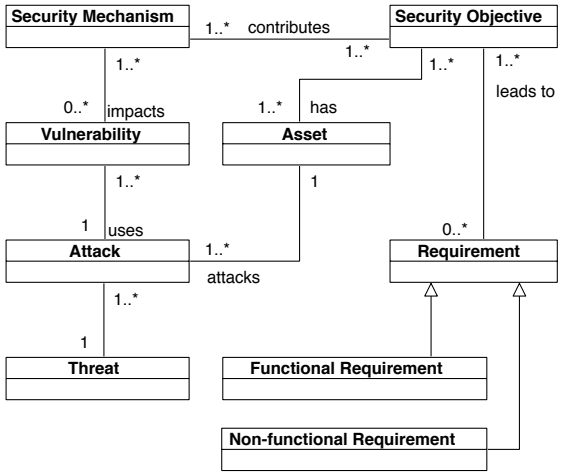

Fig. 6. Derivation of requirements

A functional security requirement specifies technical measures to achieve a security objective. In this context, the resistance of the individual mechanism chosen and the level of protection achieved against the threat-related attacks are evaluated. The chosen mechanisms are sufficient and the attack is mitigated if the combination of mechanisms used generates a level of protection that is above the attack potential attained.

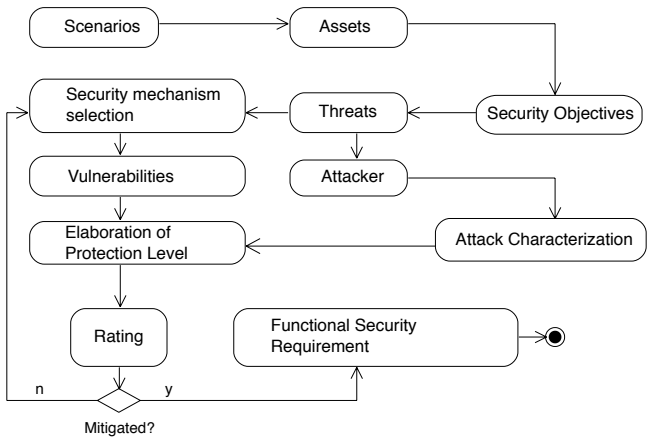

Fig. 7. Security Engineering Process

IV. Case Study – Smart Meter Gateway

In the following, the embedding of the approach in the development of a smart meter gateway (SMGW) is shown exemplary. A protection profile [45] with security requirements

has already been defined and the system undergoes a Common Criteria EAL4+ evaluation.

A. Purpose of Smart Meter Gateways

Smart Meter Gateways are decentralized communication gateways for recording and tariffing measurement data for electrical energy. Several networks are defined. The Home Area Network (HAN) provides a graphical interface for the consumer and a service interface for the service technician. The Local Metrological Network (LMN) contains the assigned counters. The gateway administrator, the billing service, and a time service communicate with the system over the wide area network (WAN). The SMGW provides a communication channel (CLS channel, Controllable Local System channel) between entities in the WAN and HAN, which is logical distinct from other communication. The security requirements for such systems are abstractly defined in a Common Criteria protection profile [45]. At the same time, the functional requirements are part of the technical guideline TR-03109 [46].

B. Need for an engineering-oriented perspective

The SMGW protection profile includes an analysis of the security problem and identifies values, assumptions, threats, security objectives, and organizational security policies to be protected. The selected level of abstraction formulates security requirements at a level that is not suitable for direct system development. All requirements must be interpreted in the context of technical guideline TR-03109. For software development, the following challenges exist when using the protection profile.

1. The level of abstraction of the security objectives is very high despite the detailing provided in Section 6 of [45] based on functional security requirements. The functional requirements remain too unspecific for the process of software development.
2. Section 3.2 identifies the need for protection for each asset. The specified protection requirement is not justified and only becomes apparent in the context of the technical guideline.
3. The representation of the threats must be in a manner appropriate to the system being developed. The representation in the protection profile is too generic. To understand the identified threats, a more detailed description and, in particular, differentiation against scenarios that are not covered by the threat, is required.
4. The attacker model defines high attack potential for the WAN attacker, whereas for the local attacker this only applies to the preparation phase. The execution of the attack is restricted to a layman. This restriction excludes attacker motivations that are outside of financial intentions (manipulated energy consumption).
5. The protection profile does not consider an important asset – the authentication data of the SMGW to the HSM. Instead, a variant attacker model is given in Section 6.3 in [46], which is in obvious contradiction to the protection profile.

The security requirements of the protection profile have been worked up in a way that creates an understanding of their motivation within software engineering. In particular, the analysis of the threats and their potential implementation through attacks leads to the traceability of security mechanisms and resulting security requirements.

C. Scenarios and Assets

Based on [45] and [46], the assets assigned to the respective perspectives were determined. The number and characterization of the assets are different from those in the protection profile. The analysis resulted in a total of 20 assets, the protection profile in total 15. Examples of differences in the determined assets are the authentication data of the consumer and the firmware itself. The authentication data of the consumer can be used to obtain information about assigned consumption data via the display interface. The IP protection of the firmware justifies the treatment as an asset. Such aspects are not taken into account in the protection profile. Examples of the assets are the configuration data of the SMGW, the authentication data of the consumer, the active firmware and the consumption data. There are threats such as unauthorized data access and unauthorized data changes, identity misuse and erroneous communication (e.g., caused by spoofing).

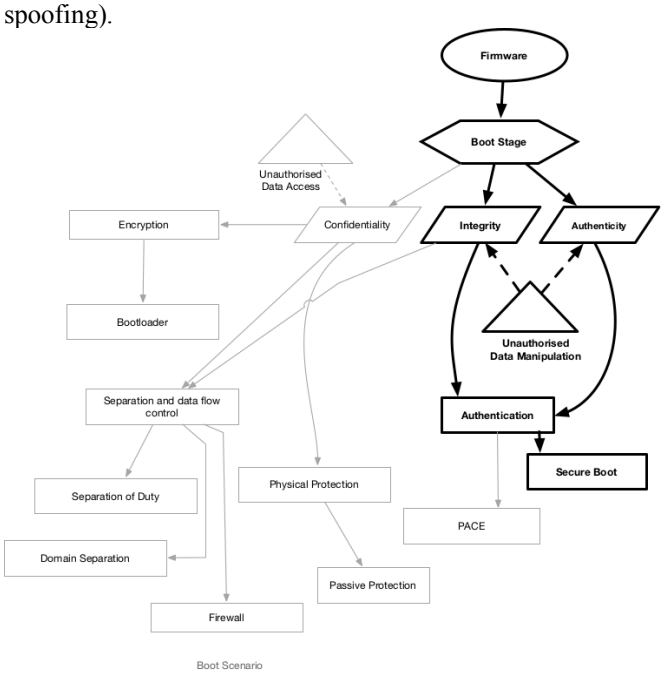

Fig. 8. Graphical Analysis for Firmware during boot

D. Threats, Attacker and Attacks

For the modelling of potential attackers, the attacker types 'remote attacker' (A_{Remote}) and 'attacker with physical access' (A_{Phys}) were defined.

TABLE III. EXEMPLARY CHARACTERIZATION OF A_{PHYS}

Factor	Description
Preparation Time	The cost-benefit analysis of the Federal Ministry of Economics and Energy [47] assumes in chapter 6, table 32, an amortisation period of a device of 13 years. It can therefore be assumed that a device will be used for a period of 13 years from the date of commissioning.
Analysis Time	Identical to preparation time
Specialized Equipment	The necessary equipment for analysing this communication consists of commercially available computers, the required software is mostly available as free software. The effort to procure the analysis equipment is classified as low to medium.
Knowledge	Most information is public. There is no effort to explore used protocols.
Expertise	Technical expert in multiple disciplines
Social capabilities	The attacker has sufficient manipulative ability to gain access to the protected HAN and LMN segments.

This is based on the assumption that attackers exist that have only the possibility for attacks via the communication interfaces. In contrast to the protection profile, a second attacker type with the possibility of physical access was defined. The characterisation of both attackers is comparable except for the possibility of physical access. In addition, the motivational profile is different, especially for the local attacker, from that chosen in the protection profile. In addition to the economic motivation (greed), the selected characterization takes into account factors such as competition / reverse engineering, curiosity, damage to the operator of the metering infrastructure and the use of the SMGWs for attacks on other infrastructure components. Overall, the potential of the local attacker is increasing in particular.

Examples of attacks are those directed against web applications. The information interface for the consumer is HTML-based. Any attacks that are relevant to web applications are also relevant to this interface. Examples include XSS attacks, injection of interpreted data (such as SQL injection), attacks from TLS connections, and exploitation of misconfigurations.

For XSS attacks, the SMGW serves as a tool to attack the consumer. As a result, further attacks can be prepared. With SQL injection (or data injection) the attacker can try to manipulate the configuration and the installed software. All variations of this scenario have in common that input data could be interpreted or executed by the SMGW without sufficient syntactic and semantic testing. Attacks on the tunnel used for data transmission are carried out by an attacker to disrupt or take over a session between the consumer display and the SMGW. Taking advantage of mismatches sums up those scenarios where a service offers an exploitable vulnerability due to improper or poor configuration. Examples include directory traversal attacks (breakout from the Web application root directory) or the possibility of unauthorized access to protected resources.

E. Security Mechanisms and Security Requirements

TABLE IV. EXEMPLARY SECURITY REQUIREMENTS

Security Mechanism	Security Requirement
Enforcement of security domains	Enforcement of security domains using SELinux. Security domains are defined along the logical communication (HAN. WAN, LMN, CLS)
Runtime protection	Use of ASLR (address space layout randomization) and SSP (stack smash protection) to mitigate impact minimization in case of successful local attack.
Minimal Deployment	There are only those applications and libraries deployed that are actually used.
Defensive Configuration	Exclusive use of secure configurations of the software components used, no experimental features.

The listed threats and attacks shown by way of example lead to the listed exemplary security mechanisms and security requirements. Within the case study, the effectiveness of the chosen security mechanisms was compared with the identified attacks and evaluated. Quantitative evaluation as an aid simplified the selection of security mechanisms and the following definition of security requirements.

V. CONCLUSION

This article shows an approach to model-based security engineering. Such an approach is required for the successful integration of security engineering into software engineering. Compared to existing methods, the presented approach

increasingly uses already described attacks to assess the effectiveness of selected security mechanisms. The abstracted level of the threat is left with this step. The method offers the possibility of a quantitative assessment and thus facilitates the design of a secure system. The process shown was successfully tested during the development of a secure communication solution for smart metering systems[1]. In further work, the previously missing tool support will be implemented. The goal is to replace the previous manual analysis process with an automated procedure. Another conceivable approach is the creation of expert systems that use methods of machine-based learning. This approach could be the subject of further research in the context of a dissertation or an industrial project.

REFERENCES

[1] Slaughter, Sandra A.; Harter, Donald E.; Krishnan, Mayuram S.: Evaluating the cost of software quality. In: Communications of the ACM 41 (1998), Nr. 8, S. 67–73

[2] Devanbu, Premkumar T.; Stubblebine, Stuart: Software Engineering for Security: A Roadmap. In: Proceedings of the Conference on The Future of Software Engineering. New York, NY, USA: ACM, 2000 (ICSE '00). – ISBN 1–58113–253–0, 227–239

[3] Haley, Charles B.; Moffett, Jonathan D.; Laney, Robin; Nuseibeh, Bashar: A framework for security requirements engineering. In: Proceedings of the 2006 international workshop on Software engineering for secure systems ACM, 2006, S. 35–42

[4] Haley, Charles B.; Laney, Robin; Moffett, Jonathan D.; Nuseibeh, Bashar: Security Requirements Engineering: A Framework for Representation and Analysis. In: IEEE Transactions on Software Engineering, 34(1) pp. 133–153 (2008)

[5] Pasquale, Liliana; Salehie, Mazeiar; Ali, Raian; Omoronvia, Inah; Nuseibeh, Bashar: On the role of primary and secondary assets in adaptive security: An application in smart grids. In: Software Engineering for Adaptive and Self-Managing Systems (SEAMS) 2012 ICSE Workshop, 4-5 June 2012, Switzerland, 2012

[6] Mellado, Daniel; Fernandez-Medina, Eduardo; Piattini, Mario: A common criteria based security requirements engineering process for the development of secure information systems. In: Computer standards & interfaces 29 (2007), No. 2, pp. 244–253

[7] Lamsweerde, Axel V.; Brohez, Simon; Landtsheer, Renaud D.; Janssens, David; Informatique, D´epartement D.: From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In: In Proc. of RHAS'03, 2003, S. 49–56

[8] Elahi, Golnaz: Security requirements engineering: state of the art and practice and challenges. Online. http://www.cs.utoronto.ca/~gelahi/DepthPaper.pdf. Version:2009

[9] Mayer, Nicolas; Rifaut, Andre; Dubois, Eric u. a.: Towards a risk-based security requirements engineering framework

[10] Chen, Yue: Software Security Economics and Threat Modeling based on Attack Path Analysis; A steakholder value driven approach, Faculty of the graduate school, University of Southern California, Diss., 2007

[11] Sheyner, Oleg M.: Scenario Graphs and Attack Graphs, School of Computer Science, Computer Science Department, Carnege Mellon University, Pittsburgh, PA, Diss., 2004

[12] Common Criteria Management Board (CCMB) (Hrsg.): Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, CCMB-2012-09-001, Version 3.1 Revision 4. 2012

[13] M. Soldal, K. S. B. Solhaug S. B. Solhaug: Model Driven Risk Analysis - The CORAS Approach. Springer Verlag Berlin Heidelberg, 2011

[14] Cysneiros, Luiz M.; Prado Leite, Julio Cesar S.: Using UML to reflect non-functional requirements. In: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research IBM Press, 2001

[15] Jürjens, Jan: Secure Systems Development with UML. Springer Verlag Berlin Heidelberg, 2005. – ISBN 3–540–00701–6

[16] Zhang, Hongyu; Zhang, Xiuzhen; Gu, Ming: Predicting defective software components from code complexity measures. In: Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on IEEE, 2007, pp. 93–96

[17] Chung, Lawrence; Prado Leite, Julio Cesar S.: On non-functional requirements in software engineering. In: Conceptual modeling: Foundations and applications. Springer, 2009, pp. 363–379

[18] Yu, Eric Siu-Kwong: Modelling Strategic Relationships for Process Reeingineering, University of Toronto, Diss., 1995

[19] Castro, Jaelson; Kolp, Manuel; Mylopoulos, John: Towards requirements-driven information systems engineering: the Tropos project. In: Information systems 27 (2002), No. 6, pp. 365–389

[20] Bresciani, Paolo; Perini, Anna; Giorgini, Paolo; Giunchiglia, Fausto; Mylopoulos, John: Tropos: An agent-oriented software development methodology. In: Autonomous Agents and Multi-Agent Systems 8 (2004), No. 3, pp. 203–236

[21] Garzetti, Maddalena; Giorgini, Paolo; Mylopoulos, John; Sannicolo, Fabrizio: Applying Tropos Methodology to a real case study: Complexity and Criticality Analysis. (2002)

[22] Cares, Carlos; Franch, Xavier; Mayol, Enric: Extending tropos for a prolog implementation: A case study using the food collecting agent problem. In: International Workshop on Computational Logic in Multi-Agent Systems Springer, 2005, pp. 396–405

[23] Morandini, Mirko; Nguyen, Duy C.; Perini, Anna; Siena, Alberto; Susi, Angelo: Tool-supported development with tropos: The conference management system case study. In: International Workshop on Agent-Oriented Software Engineering Springer, 2007, pp. 182–196

[24] Hadar, Irit; Kuflik, Tsvi; Perini, Anna; Reinhartz-Berger, Iris; Ricca, Filippo; Susi, Angelo: An empirical study of requirements model understanding: Use Case vs. Tropos models. In: Proceedings of the 2010 ACM Symposium on Applied Computing ACM, 2010, pp. 2324–2329

[25] Object Management Group: UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification, Version 1.1. 2008

[26] Ericson, Clifton A.; Ll, Clifton: Fault tree analysis. In: System Safety Conference, Orlando, Florida, 1999, pp. 1–9

[27] Roberts, N.H.; Vesely, W.E.; Haasl, D.F.; Gold- berg, F.F.: Fault Tree Handbook (NUREG-0492). In: US Nuclear Regulatory Commission (1981)

[28] Fredriksen, Rune; Kristiansen, Monica; Gran, Bjørn A.; Stølen, Ketil; Opperud, Tom A.; Dimitrakos, Theo: The CORAS framework for a model-based risk management process. In: International Conference on Computer Safety, Reliability, and Security Springer, 2002, pp. 94–105

[29] Stamatelatos, Michael; Vesely, William; Dugan, Joanne; Fragola, Joseph; Minarick, Joseph; Railsback, Jan: Fault tree handbook with aerospace applications. 2002

[30] Shostack, Adam: Experiences threat modeling at microsoft. In: Modeling Security Workshop. Dept. of Computing, Lancaster University, UK, 2008

[31] S. Lipner: "The Trustworthy Computing Security Development Lifecycle". Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04) Tucson 2004

[32] Myagmar, Suvda; Lee, Adam J.; Yurcik, William: Threat modeling as a basis for security requirements. In: Symposium on requirements engineering for information security (SREIS) Bd. 2005 Citeseer, 2005, pp. 1–8

[33] Massacci, Fabio; Mylopoulos, John; Zannone, Nicola: Computer- aided support for secure tropos. In: Automated Software Engineering 14 (2007), No. 3, pp. 341–364

[34] Mouratidis, Haralambos; Giorgini, Paolo: Secure tropos: a security-oriented extension of the tropos methodology. In: International Journal

[1] OpenLimit SignCubes AG and Power Plus Communications AG developed a TR-03109 compliant Smart Meter Gateway (SMGW). This Gateway is currently under Common Criteria certification.

of Software Engineering and Knowledge Engineering 17 (2007), No. 02, pp. 285–309

[35] Giorgini, Paolo; Mouratidis, Haralambos; Zannone, Nicola: Modelling security and trust with secure tropos. In: Integrating Security and Software Engineering: Advances and Future Vision (2006), pp. 160–189

[36] Jürjens, Jan: UMLsec: Extending UML for secure systems development. In: ≪UML≫ 2002—The Unified Modeling Language (2002), pp. 1–9

[37] Lodderstedt, Torsten; Basin, David; Doser, Jürgen: SecureUML: A UML-based modeling language for model-driven security. In: International Conference on the Unified Modeling Language Springer, 2002, pp. 426–441

[38] Brucker, Achim D.; Doser, Jürgen; Wolff, Burkhart: A model transformation semantics and analysis methodology for SecureUML. In: International Conference on Model Driven Engineering Languages and Systems Springer, 2006, pp. 306–320

[39] Cenys, A; Normantas, A; Radvilavicius, L: Designing role-based access control policies with UML. In: Journal of Engineering Science and Technology Review 2 (2009), pp. 48–50

[40] Matulevicius, Raimundas; Dumas, Marlon: A Comparison of SecureUML and UMLsec for Rolebased Access Control. In: Proceedings of the 9th Conference on Databases and Information Systems, 2010, pp. 171–185

[41] Matulevicius, Raimundas; Dumas, Marlon: Towards model transformation between SecureUML and UMLsec for role-based access control. In: Databases and Information Systems VI, IOS Press (2011)

[42] Common Criteria Management Board (CCMB) (Hrsg.): Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components, CCMB-2012-09-002, Version 3.1 Revision 4. 2012

[43] Common Methodology for Information Technology Security Evaluation, Evaluation methodology, CCMB-2012-09-004, Version 3.1 Revision 4. 2012

[44] Armin Lunkeit, Modellbasiertes Security-Engineering in der Softwareentwicklung, PhD Thesis, TU Berlin, 2018

[45] Bundesamt für Sicherheit in der Informationstechnik (BSI) „Protection Profile for the Gateway of a Smart Metering System (Smart Meter Gateway PP) Schutzprofil für die Kommunikationseinheit eines intelligenten Messsystems für Stoff- und Energiemengen Version 1.3“, 2014.

[46] Bundesamt für Sicherheit in der Informationstechnik (BSI) „Technische Richtlinie BSI TR-03109-1 Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystems“, Version 1.0. 2013.

[47] Ernst & Young: Kosten-Nutzen-Analyse für einen flächendeckenden Einsatz intelligenter Zähler, 30.07.2013. 2013