Security Risk Assessments: Modeling and Risk Level Propagation

DANIEL ANGERMEIER, HANNAH WESTER, KRISTIAN BEILKE, and GERHARD HANSCH, Fraunhofer-Institute AISEC, Germany
JÖRN EICHLER, Freie Universität Berlin, Institute of Computer Science, Germany

Security risk assessment is an important task in systems engineering. It is used to derive security requirements for a secure system design and to evaluate design alternatives as well as vulnerabilities. Security risk assessment is also a complex and interdisciplinary task, where experts from the application domain and the security domain have to collaborate and understand each other. Automated and tool-supported approaches are desired to help manage the complexity. However, the models used for system engineering usually focus on functional behavior and lack security-related aspects. Therefore, we present our modeling approach that alleviates communication between the involved experts and features steps of computer-aided modeling to achieve consistency and avoid omission errors. We demonstrate our approach with an example. We also describe how to model impact rating and attack feasibility estimation in a modular fashion, along with the propagation and aggregation of these estimations through the model. As a result, experts can make local decisions or changes in the model, which in turn provides the impact of these decisions or changes on the overall risk profile. Finally, we discuss the advantages of our model-based method.

CCS Concepts: • **Software and its engineering** → **Risk management**; • **Security and privacy** → **Security requirements**; **Software security engineering**; *Usability in security and privacy;*

Additional Key Words and Phrases: Security risk assessment, risk analysis, security engineering, model-based, secure design, threat modeling

ACM Reference format:
Daniel Angermeier, Hannah Wester, Kristian Beilke, Gerhard Hansch, and Jörn Eichler. 2023. Security Risk Assessments: Modeling and Risk Level Propagation. *ACM Trans. Cyber-Phys. Syst.* 7, 1, Article 8 (February 2023), 25 pages.
https://doi.org/10.1145/3569458

1 INTRODUCTION

Security risk assessment (SRA) is a crucial part of requirements engineering and enables the systematic deduction of security requirements. Additionally, **Security risk assessment** (**SRA**) supports the prioritization and execution of further security-related tasks in the engineering life

Authors' addresses: D. Angermeier, H. Wester, K. Beilke, and G. Hansch, Fraunhofer-Institute AISEC, Lichtenbergstraße 11, Garching, Bavaria, Germany, 85748; emails: daniel.angermeier@aisec.fraunhofer.de, hannah.wester@aisec.fraunhofer.de, kristian.beilke@aisec.fraunhofer.de, gerhard.hansch@gmail.com; J. Eichler, Freie Universität Berlin, Institute of Computer Science, Takustr. 9, Berlin, Berlin, Germany; email: joern.eichler@fu-berlin.de.

2378-962X/2023/02-ART8
https://doi.org/10.1145/3569458

cycle. Therefore, **SRA**s represent mandatory steps in many regulations and international standards (e.g., Reference [34] as well as Reference [17] for the automotive domain).

However, risk assessment presents a challenging task of high complexity, as all possible interactions (not only all *specified* interactions) with the **system under development (SUD)** may result in a violation of protection needs. Consequently, it is advisable to support the analyst with a computer-aided, model-based approach to avoid omissions or other "human errors" and master complexity.

Generally, data for a high grade of computer aid for **SRA** is often missing: Defining functional and structural properties of a system is an integral part of the system's development, while modeling potential misuses is not. System models are designed with a given purpose in mind. As "all models are wrong, but some are useful" [5], existing models from system development only partially fit the purpose of security assessment. Some aspects may be over-represented, while others may be covered insufficiently. Therefore, we follow the hypothesis that a dedicated model for the security risk assessment creates a better match for the task. Nevertheless, a dedicated **SRA** model still represents a simplification of the **system under development** (**SUD**). This simplification is useful as it reduces complexity. Attackers, however, interact with existing systems. Thus, attackers are not affected by the model's limitations. Therefore, human expertise by a security analyst remains a necessity for proper **SRA**s to explore the attacker's options with human creativity.

Our graph-based modeling approach aims to achieve the best of both worlds. While the full graph provides a rich set of relations between the elements of the **SRA** for computer-aided functionality, selected parts of the graph can also be visualized to supplement human experts. They further support the identification of critical paths and the derivation of effect chains from the model while keeping the human analyst in the loop. For example, the connections between components can be derived from the dataflows and visualized in a diagram. Likewise, attack paths in the graph can be visualized similar to attack trees. Graphical representations also support the communication among developers, security experts, and other stakeholders (e.g., management), making the analysis comprehensible and verifiable for all involved stakeholders.

This article is based on previous work and practical experiences in the automotive field (cf. References [2, 4, 8]) and extends the work presented in Reference [1]. We provide the following contributions:

- Refinement of an extended metamodel for security risk assessments to represent complex dependencies based on the **SUD** between security goals, threats, controls, and assumptions
- Definition of basic formal properties to evaluate these dependencies
- Ruleset for risk value calculation for all relevant elements

Benefits from our contributions include globally calculated estimations concerning the risk value for all relevant elements. Thus, critical security goals, threats, assumptions, and controls can be identified and the consequences of design alternatives can be evaluated. Complementary to the effects of controls and assumptions on attack feasibility, our contributions provide means to represent and evaluate effects on security goals' protection needs.

The remainder of this article is organized as follows. After discussing related work in Section 2, we provide background information on the used risk assessment method in Section 3. We then present and demonstrate our graph-based modeling approach in Section 4, and describe the information flow and risk calculation in Section 5, before we conclude in Section 6.

2 RELATED WORK

To cope with the increased exposure to cyber-attacks, as demonstrated in Reference [25], harmonized regulations and international standards establish **SRA**s as a mandatory activity of future

development processes (cf. References [17, 34]). The mandatory implementation of security risk assessment and management processes into the development, production, and post-production phases requires suitable methods. Generally, model-based risk assessment during development includes creating a model of the **SUD**, an impact assessment, and a threat assessment.

An overview of 18 different security requirements engineering approaches and techniques, including CORAS, Misuse Cases, Secure Tropos, and UMLSec, is provided by Fabian et al. [9]. Gritzalis et al. [12] compare validity, compliance, costs, and usefulness of popular risk assessment methods including EBIOS, MEHARI, OCTAVE, IT-Grundschutz, MAGERIT, CRAMM, HTRA, NIST SP800, RiskSafeAssessment, and CORAS. Most of these approaches do not provide a formal method in combination with a dedicated metamodel. One exception here is CORAS [22], a model-driven risk assessment method using a graphical notation applying a domain-specific language. The models are analyzed manually by human analysts, as CORAS opts for a dedicated graphical concrete syntax and does not define alternative representation, often preferred for larger models (cf. Reference [21]).

A dedicated survey on attack and defense modeling approaches utilizing directed acyclic graphs is provided by Kordy et al. [19]. The surveyed approaches feature a focused perspective on security-specific properties and allow for calculations, e.g., critical attack paths. However, they do not entail an integrated perspective of the **SUD**, and its security properties aligned with typical artifacts from the development phase. According to the classification of Reference [19], we present a defense-oriented approach that combines static parts for the system model with sequential parts for the risk analysis. Furthermore, we use a **directed acyclic graph (DAG)** for general security modeling and quantitative assessment, including conceptual, and quantitative extensions in a semi-formal way. Parts and different iterations are implemented by commercial tools and applied by independent users to perform realistic assessments.

A state-of-the-art threat assessment method and basis for many cybersecurity risk assessment methodologies is STRIDE, standing for the six considered threat classes Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege that can be used to threaten security objectives [18]. Each threat class antagonizes at least one security property. Spoofing a false identity violates the authenticity property of entities, tampering threatens the integrity of data and processes, repudiating a responsibility interferes with non-repudiation, e.g., of process interaction; information disclosure the confidentiality of data and processes; denial of service the availability of components, data, and processes; and elevation of privileges enables the unauthorized execution of actions. A prominent refinement of STRIDE is STRIDE-per-Element, which considers that certain threats are more prevalent with certain elements of a model, which facilitates threat identification in general by focusing on the most relevant threats [31]. Combining STRIDE with attack trees is used by several recent security risk analysis frameworks, e.g., References [13, 26].

Bayes attack graphs are one method to assess security risks in IT networks and assess vulnerabilities, enabling Bayesian inference procedures for general attack graphs (cf. References [19, 28, 37]). In the concept and development phase, whose support is in the focus of the method presented here, there are usually no known vulnerabilities and only rarely weaknesses. Thus, a percentage value required for Bayesian networks cannot be directly determined and, in practice, usually leads to intense disagreement among the respective responsible parties. To evaluate the technical difficulty for an attacker to execute an attack, we instead use a qualitative scheme such as the Common Methodology for Information Technology Security Evaluation [6] to rate the required capabilities for each attack step.

A way to avoid the common problem of inconsistencies within **SRA**s is using an appropriate ontology. A good overview of risk assessment ontologies is provided by Souag et al. [33]. The authors

compare many existing ontologies and create a new one, including additional high-level security concepts, axioms, and attributes. They generally specify their models in the Web Ontology Language and apply a certain level of automation with queries using the Semantic Query-enhanced Web Rule Language (cf. Reference [27]). Their target audience is requirements engineers and thus is not focused on risk assessment. In contrast to this high-level approach, we provide many more details concerning the structure, relations, and propagation of ratings within the **SRA**. A similar approach is applied in Reference [35] for the automated search for known vulnerabilities in incomplete or inconsistent described systems. An approach that deducts recommendations for high-level security measures from assessed security risks can be found in Reference [14]. Unlike our proposal, the presented risk analysis method and metamodel provide only a limited security risk evaluation. Such approaches might significantly benefit from the here presented metamodel and methods.

Computer Aided Integration of Requirements and Information Security (CAIRIS) is a framework to manage security, usability, and design artifacts. Reference [10] achieves a certain level of automation and visualization. The framework's aim is much broader in scope, as it also encompasses usability and requirement engineering activities. Regarding security risk analysis, the authors propose a broad ontology of concepts. CAIRIS expresses similar information compared to our approach. However, the implementation[1] includes many concepts that keep additional information but adds effort for documentation, maintenance, and consistency (i.e., environments, vulnerabilities, obstacles, use cases). Details and conciseness of core concepts for **SRA**s (goals, threats, controls, and their interactions) seem to be impacted by the breadth of the approach. Similarly, we consider separating the security and the (functional) development domains to benefit tailored application.

A combination of UML-based information system modeling with Bayesian attack graphs for assessing attack probabilities are CySeMol [32] and its extension P^{2}CySeMoL [15]. The relational model and the thereupon built inference engine allow for evaluating "what-if" scenarios. Networks consisting of well-known components can be evaluated efficiently due to the predefined granularity of the components. While this approach enables modifications of the model during analysis, it does not support iterative dissection or damage transformations and hardly copes with new components. A further approach to formally describe security properties in a security risk analysis framework, based on model-checking and a Markov decision process do to determine risk probabilities, is presented in Reference [23].

A proprietary framework for information risk analysis is the FAIR approach in Reference [11]. It includes a taxonomy, a method for measuring the driving risk factors, and a computational engine to simulate relationships between these factors. Key factors in determining risks are the *Loss Event Frequency*, based on the *Threat Event Frequency* and the *Vulnerability*, and the *Loss Magnitude*, reflecting the impact. Due to the dependency on measurable and historical factors, initial risk assessments and non-metric environments pose severe problems for users. In contrast to the presented approach that focuses on the overall impact, FAIR is limited to risks for information assets.

A combination of the automotive Hazard Analysis And Risk Assessment with STRIDE, intended to support the functional concept phase by a straightforward quantification of the impacts of threats and hazards, is the **Security-Aware Hazard And Risk Analysis (SAHARA)** method [24]. Similarly, the conventional Failure Mode Effects Analysis is extended by vulnerabilities by the FMVEA method [30] and focuses on the technical concept phase of the development. While both these automotive-oriented methods rely on a model of the **SUD**, they use a top-down assessment approach, about by what a specific threat, respectively, safety hazard might be caused. Furthermore,

[1] https://cairis.org/.

in their assessment, they do not consider interactions, respectively, sequences, and the effects of security measures and of their propagation along the model.

A combination of FAIR, SAHARA, and FMVEA is the probabilistic **Risk-Tree Based Method for Assessing Risk in Cyber Security (RISKEE)** approach [20]. As the long form of the name indicates, it combines risk calculation with attack trees. Based on FAIR, the considered risk factors are frequency, vulnerability, and magnitude of vulnerabilities. A specialty of the RISKEE approach is the relation and visualization of the calculated and the acceptable risk as a loss-exceeding curve.

Popular commercial tools for threat modeling are the Microsoft Threat Modeling Tool[2] and the fortisee SecuriCAD,[3] which also includes probabilistic attack simulation. They both provide a graphical interface for modeling current and abstract IT environments and assessing potential security issues. While the Threat Modeling Tool utilizes a STRIDE-based risk assessment method, SecuriCAD also supports evaluating possible attack vectors by Monte Carlo simulation. Both tools regard coarse-grained attack paths focusing on cloud and enterprise IT but lack attack feasibility factor propagation or damage transformation. Commercial tools are currently also developed to support **SRA**s in the automotive domain, including the Yakindu Security Analyst[4] and Ansys Medini Analyze.[5]

3 BACKGROUND

The risk assessment approach used in this article is based on the **Modular Risk Asessment (MoRA)** method (cf. References [7, 8]). Figure 1 depicts four core activities of the method framework: "Model the Target of Evaluation," "Determine Protection Needs," "Analyze Threats," and "Analyze Risks." The first step decomposes the **SUD** into relevant functions, data elements, components, and dataflows. The next step identifies *security goals* as combinations of assets detailed in the **SUD** and their required security properties. The third step identifies threats to the assets, analyzing systematically elements of the **SUD**. Additionally, actual or proposed controls can be added to mitigate identified threats. Impact and attack feasibility ratings for security goals as well as threats and controls are estimated in the last step. Risk levels are derived from those estimations. For more details on the application of **Modular Risk Assessment** (**MoRA**), we refer to aforementioned publications and [3, 4]. While this work is based on **MoRA**, we align the terminology in this article with ISO/SAE 21434.

MoRA relies on an *assessment model* and *catalogs* to homogenize assessments within a common application domain. Thus, the assessment model and the catalogs represent a common ground for all stakeholders regarding core aspects of risk assessments like evaluation criteria or threat classes. Note that standards and regulations can suggest or define parts of the assessment model, such as the threat model given in Annex 5 of the *R-155 UN Regulation on uniform provisions concerning the approval of vehicles with regard to cybersecurity and of their cybersecurity management systems* [36].

Our graph-based modeling approach as specified in extension to our previous work in Section 4 augments **MoRA**'s representation. It facilitates the method implementation and is based on experience from several years of practical application in industrial development projects. The model and calculation rules are the basis for tooling, such as the Yakindu Security Analyst. While we present our generic metamodel without a specific syntax in this article, we successfully adapted the metamodel and tooling to accommodate the specific requirements for risk assessments in standards and regulation, such as ISO/SAE 21434 (cf. Reference [17]) or the IEC 62443 (cf. Reference [16]).

[2] https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool.
[3] https://www.foreseeti.com/securicad/.
[4] https://www.itemis.com/de/yakindu/security-analyst/.
[5] https://www.ansys.com/products/systems/ansys-medini-analyze-for-cybersecurity.

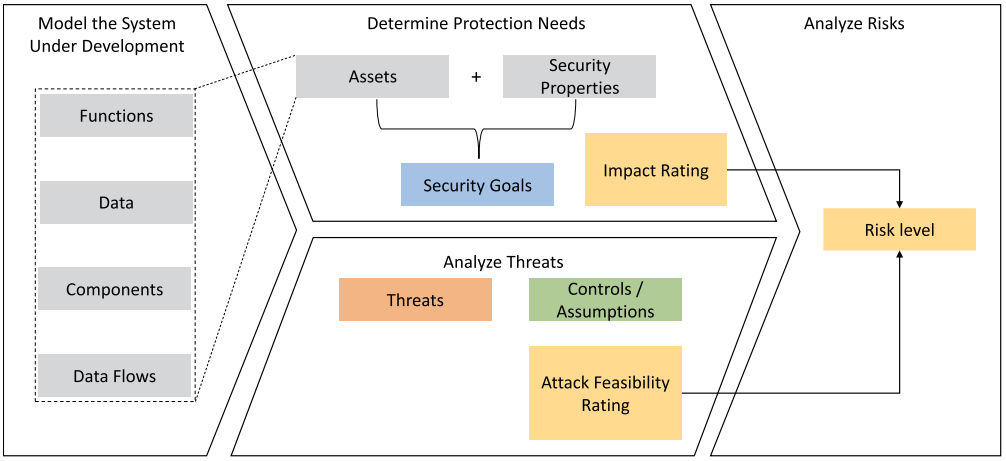

Fig. 1. Main activities and core concepts in security risk assessments according to **MoRA**. Section 4 provides more details on the metamodel representing these core concepts.

4 METAMODEL

The following section presents the metamodel of our **SRA** model previously introduced in Reference [1]. It encompasses a focused representation of the **SUD** itself as well as risk assessment-specific core concepts like security goals, damage scenarios, threats, controls, and assumptions. Providing all these elements in one model allows for derivation and validation of relations between and properties of elements of the **SRA** aligned with the **SUD** (cf. Reference [4]). This facilitates comprehension and traceability. The core concepts are presented along **MoRA**'s main activities followed by a modeling example.

4.1 Model the System under Development

The **SUD** model serves as the foundation for the analysis. It includes assets, which are required to understand the protection needs and potential damage scenarios. The **SUD** model also provides an overview of potential interactions with the **SUD**. This facilitates the elicitation of potential threats against it. Furthermore, by modeling the **SUD** in cooperation with domain experts, security analysts gain a solid understanding of the **SUD**. Likewise, as all arguments are rooted in the system model, explaining risks to domain experts is facilitated. Note that every risk assessment requires an abstraction of the analyzed item, i.e., a model, in the analyst's mind. Documenting the model and discussing it with domain experts improves the model's correctness in our experience.

The graph of the **SUD** consists of four sets, visualized as nodes, and the relations between these nodes, connecting them as edges, as shown in Figure 2. The four sets of nodes in the risk assessment graph represent the *functions*, *data elements*, *components*, and *dataflows* of the **SUD**. Within each of these sets, the subelement relation ("is subdata of," "is subcomponent of," and "is subfunction of") represents a hierarchy between the elements. A component, for example, can be refined into its sub-components (e.g., the component "vehicle" has subcomponents "brake ECU" and "airbag ECU" and "brake ECU" could consist of a subcomponent "software platform").

Dataflows each have a sender and a receiver, resulting in a matching "has sender" and "has receiver" relation from the dataflow to the sender and receiver component. Furthermore, the dataflow has a "transmits" relation to one or more data elements. Note that the metamodel allows components that neither receive nor transmit data if required. Components have a "stores" relation to locally stored data elements. This is mainly used for data that might never be transmitted, such

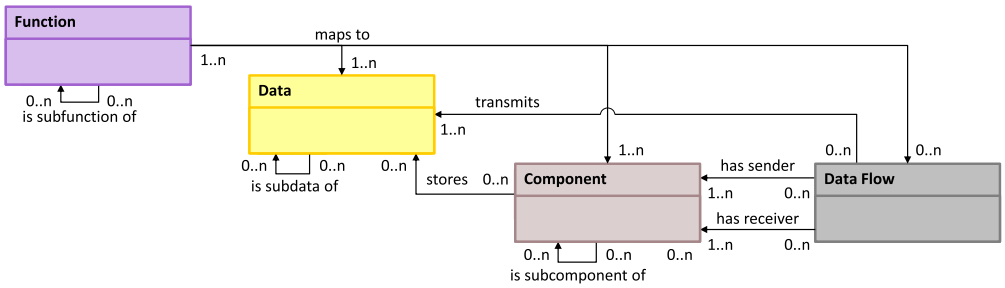

Fig. 2. Metamodel including functions: Functions are implemented using data, components, and dataflows.

as private keys for cryptographic operations. All relations between components and data are non-exclusive, i.e., components can send, store, and receive the same data element as other components. These relations are depicted in Figure 2.

Based on these explicit relations, implicit relations can be derived: for each sender, a "produces" relation to the sent data elements, and for each receiver a "consumes" relation to received data elements. Therefore, interface definitions of components can be derived from the dataflow definitions. These implicit relations are always calculated from the existing dataflow definitions and never defined explicitly to avoid inconsistencies.

Functions have a "maps to" relation to data elements, components, and dataflows, as shown in Figure 2. These relations imply that functions are implemented by data processing and transmission, which in turn are executed by components. The functions thus depend on their mapped elements.

We have chosen this representation for the **SUD** as it fully supports **SRA**s based on **MoRA** [8] while it also captures only information typically created during system development. For example, an **SUD** provided as a set of UML use case diagrams, component diagrams with information flows, and corresponding class diagrams can be used as input for the modeling activity. The functions can be extracted from use case diagrams, components and dataflows from component diagrams, and data elements from class diagrams. Thus, well-established modeling languages can be used as input for the first step "Model the Target of Evaluation." Furthermore, the information captured in the SRA model can be "translated" back to UML with low effort, improving the communication between domain and security experts. Consequently, our **SUD** representation supports the mutual understanding and the collaborative creation of the **SRA** model by all stakeholders, maintaining an unambiguous reference for the following steps of the risk assessment.

4.2 Determine Protections Needs

The protection needs are captured through the risk assessment-specific core concepts security goals and damage scenarios.

Security Goals (SG) define security properties for assets, where "security property" denotes an asset's property, such as confidentiality, availability, or integrity. For the sake of simplicity, we focus on these three security attributes, but the method may be extended to any set of properties. Assets are modeled as elements of the **SUD**, reflected by the "is asset of" relation, as shown in Figure 4. For example, a medical system stores the data element "patient data." "Confidentiality of patient data" represents a security goal with the security property "confidentiality" and the asset "patient data."

Note that our definition of "security goal" diverges from the similar term "cybersecurity goal" as defined in ISO / SAE 21434 [17]. As the standard does not provide a compact term for "cybersecurity

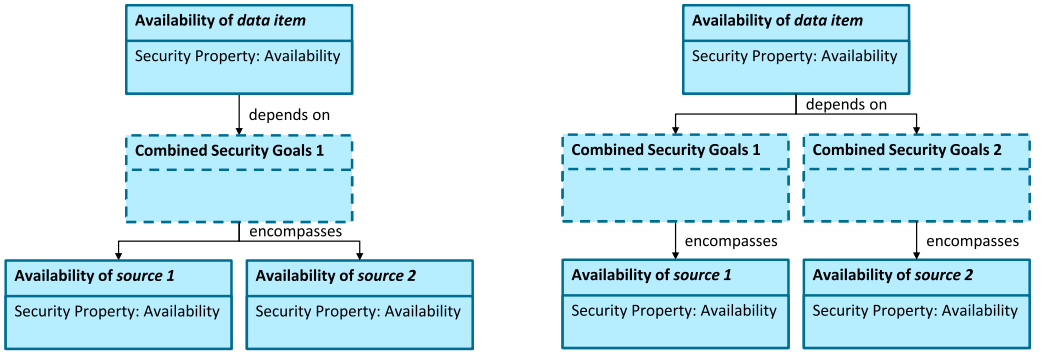

Fig. 3. Dependencies of security goals. The left side shows how the Availability of *data item* can be violated only if the Availability of *source 1 and source 2* are not given. The right side depicts a case where violating a single dependency is sufficient to violate the security goal at the top.

property of asset," we stick to "security goal," remaining consistent with our previous publications on the same topic.

Violation of a relevant security goal leads to one or more *damage scenarios*. This is denoted by the "violation causes" relation. A damage scenario is defined by a non-empty set of impact criteria. In our example, violation of the security goal "Confidentiality of patient data" causes the damage scenario "Unauthorized access to personal data," which entails the impact criterion "Substantial violation of laws" as an attribute. Impact criteria are part of the assessment model. The assessment model assigns an impact rating to each criterion. Impact criteria can be structured in impact categories, such as safety, financial, operational, or privacy. These four impact categories are required by Reference [17] and were previously proposed in, e.g., Reference [29]. The assessment model with the impact categories and corresponding impact criteria is adaptable to organizations and their field of operation.

Security goals might depend on other security goals. For example, the availability of a function depends on the availability of a component executing the function. If the availability of the component is violated, then the availability of the function is also violated. Consequently, the second security goal depends on the first.

These dependencies can be independent of each other or require several dependencies to be violated. For example, if two independent sources provide a data item, then the security goal "Availability of data item" is violated only if the security goals "Availability of the first source" *and* "Availability of the second source" are both violated. For the graphical representation of this example, see Figure 3.

We introduce the element "Combined Security Goals" to define these dependencies. A security goal depends on an arbitrary number of mutually independent "Combined Security Goals" nodes. Each "Combined Security Goals" node then relates to one or more security goals.

Note that arbitrary logical expressions with AND and OR, as often seen in classical attack trees, can always be transformed into disjunctive normal form to fit this metamodel, including specific sequences of attacks.

4.3 Analyze Threats

The threat analysis is captured through the risk assessment-specific core concepts of threats, controls, and assumptions.

Security goals are threatened by combinations of *threats*, as depicted in Figure 4. Similarly to the dependency on other security goals, threats can either threaten a security goal independently

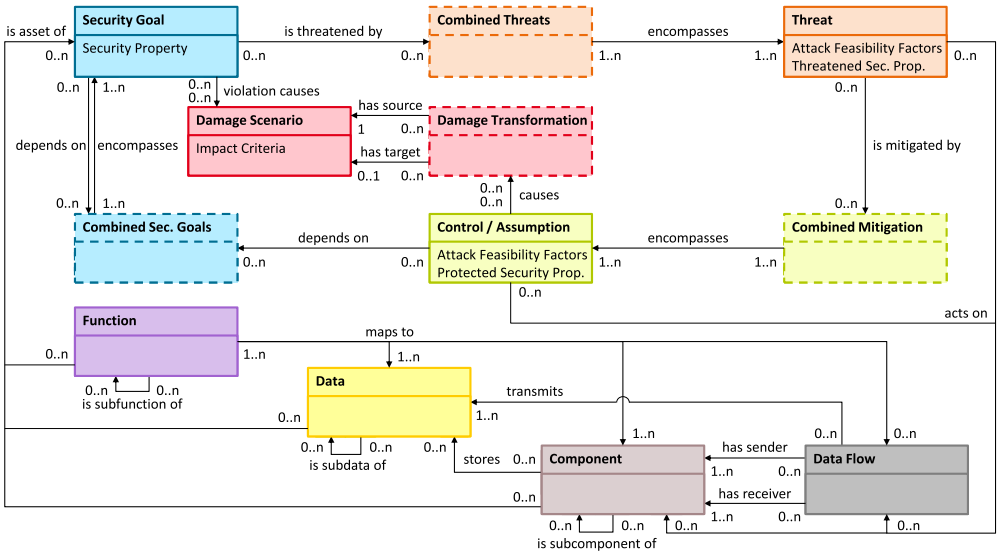

Fig. 4. Complete metamodel including controls and assumptions: Threats can be mitigated by controls, assumptions, or combinations of these.

of each other or require other threats to also execute successfully. For example, the integrity of a function can be threatened by eavesdropping on a message as attack preparation *and* by subsequently replaying the eavesdropped message. We introduce the element "Combined Threats" to define these dependencies. A security goal is threatened by an arbitrary number of mutually independent "Combined Threats" nodes. Each "Combined Threats" node then relates to one or more threats.

Threats provide the following attributes:

- *Attack feasibility factors* help to estimate the *attack feasibility rating* to realize the threat. The attack feasibility factors themselves are defined in the assessment model and, therefore, can be adapted to any standard or organizational needs. For example, CEM [6] defines five attack feasibility factors for the estimation of the "required attack potential," i.e., *Elapsed Time*, *Expertise*, *Knowledge of the TOE*, *Window of Opportunity*, and the necessary *Equipment*, along with a set of predefined values (e.g., "Layman") and corresponding numeric values for each attack feasibility factor.
- *Threatened security properties* defines the security properties a threat might violate. For example, the threat "information disclosure" threatens the security property "confidentiality."

Threats act on the physical manifestation of the **SUD**. The physical aspects of the **SUD** are modeled as components and dataflows (including wireless transmissions) as described in Subsection 4.1, while data and functions are processed by these elements. The "acts on" relation is not required to model a risk assessment but useful to help human analysts understand the threats. Additionally, the model of the **SUD** combined with the threatened security properties can be used to identify and validate potentially violated security goals. For example, it is plausible to assume that a threat "information disclosure" on a dataflow threatening the confidentiality of the transmitted data items affects the confidentiality of functions mapped onto the data items. Vulnerabilities are not represented with a metamodel element. In the context of MoRA, a threat causing a relevant risk that is not mitigated poses a vulnerability of the **SUD**.

(Cybersecurity) controls and assumptions mitigate threats. Combinations of these elements are represented by a "Combined Mitigations" node type. In some cases, controls and assumptions may be similar due to technical circumstances. Technical measures like channel encryption used by the **SUD** are usually modeled as controls. By contrast, laws of nature, responsibilities or controls of third parties, attacker capabilities, and the analysis limits are documented as assumptions. Controls and assumptions, similar to threats, provide attack feasibility factors and protected security properties. The attack feasibility factors facilitate the estimation of the control's or assumption's effect on the attack feasibility of related threats. Section 5 provides details on how to combine the attack feasibility factors in the **SRA** model.

Additionally, controls as well as assumptions may cause changes in the impact of the violation of security goals modeled as damage transformation. In this case, a damage scenario is replaced by another damage scenario or entirely removed by assigning no transformation target.

For example, suppose a valve in a factory is controlled by network messages. In that case, an attacker might manipulate these messages to violate the security goal "integrity of valve control" and cause the damage scenario "explosion of a pressure tank." The control "opening the valve on locally measured high pressure" cannot prevent this manipulation, but effectively transforms the (source) damage scenario into the less critical target damage scenario "production outage."

Assumptions may be used to bring information into the model that has not been explicitly modeled in the **SUD** but is important in its effect on the analysis, such as limitations of the assumed attacker model. Unlike controls, assumptions do not depend on security goals.

MoRA also supports catalogs for threat and control classes (cf. Reference [8]). These classes may entail a pre-assessment of attack feasibility factors, estimating the attack feasibility to execute the threat or break the control. The threats and controls in the **SRA** can use these pre-assessed values but also override them to reflect the more specific context of the **SRA**. In addition to providing a common ground for security risk assessments, these catalogs also support the analyst in "not overlooking" known threats and validating conformity to regulatory prescribed threat and control catalogs.

Controls are implemented by the **SUD** and may thus depend on its security goals. For example, the control "digital signature" requires a component to create the signature and another component to check its validity. Consequently, instead of breaking the signature, an attacker can try to violate the security goal "confidentiality of the private key" on the signing component or the security goal "integrity of the certificate" on the component executing the signature check. Both attacks can circumvent the control "digital signature." We model this by introducing a dependency of controls on security goals or combinations of security goals, again using the "Combined Security Goals" node type. Consequently, impacts caused by the loss of confidentiality of cryptographic keys do not have to be estimated directly but are reflected by additional attack paths on controls. Therefore, if the impact rating for security goals of the protected functions, data items, or components changes, then this change is consistently reflected for the risks caused by attacks on cryptographic keys.

4.4 Analyze Risks

Figure 4 depicts the full metamodel with all node types and their relations to each other. We do not model risks as separate elements. Instead, risk levels can be determined for every security goal, threat, control, or damage scenario as described in Section 5.

A risk level is determined by the combination of potential damages (impact rating) and the attack feasibility rating to cause these damages. The impact rating is determined by the impact criteria originating from the damage scenarios related to a risk. The attack feasibility rating is determined by the attack feasibility factors of the threats and the attack feasibility factors of the controls mitigating them.

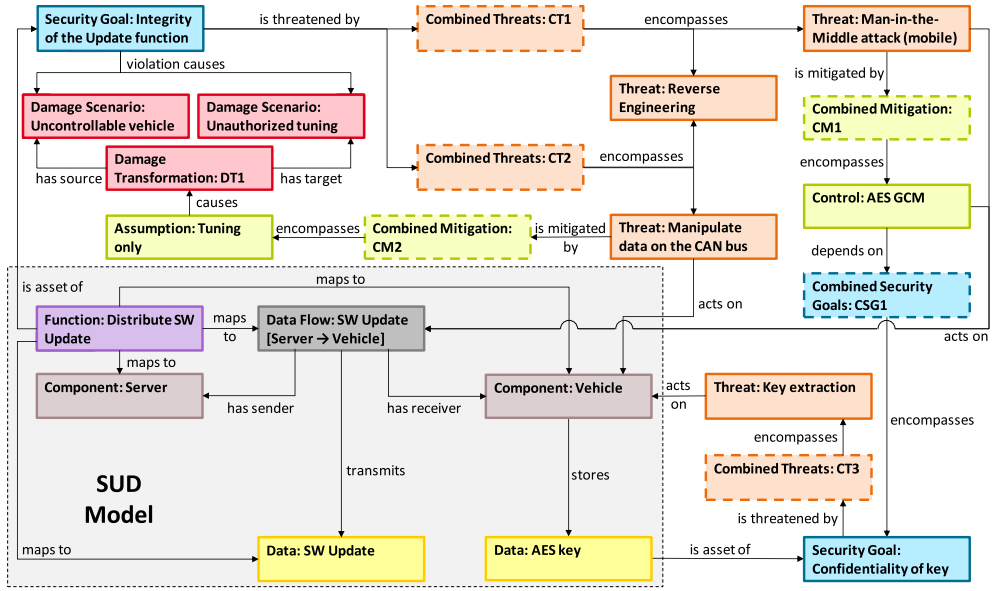

Fig. 5. This example shows an excerpt of an instance of our graph for a risk assessment of a fictitious software update function.

In our practical experience, this model of the **SUD**, represented by functions, data, components, and dataflows, is well suited for **SRA**s and easy to understand for system developers. All nodes in the risk assessment core concept relate to this model. Security goals are properties of the **SUD**. Threats and controls act on the **SUD**, modeling the interaction with the system. As outlined in Reference [3], the relations between the risk assessment elements can be validated by tracing them back to the model of the **SUD**. Similarly, Reference [3] provides a method to propose new nodes and relations based on the model of the **SUD**. Consequently, the creation of risk assessment elements can partially be automated, requiring the analyst to check and modify the proposals and to specify proposed elements further.

4.5 Modeling Example

Figure 5 depicts an instance of a metamodel for a fictitious software update function. Note that the elements in the risk assessment and their relations can be defined without actually providing a graphical representation, e.g., in a tabular representation. This is important, as a full graph for a complete risk assessment possesses high complexity, owing to a large number of nodes with many relations between them. Thus, a complete graphical representation is typically difficult to process for a human analyst. However, plotting selected parts graphically is helpful in our experience. Consequently, we chose a small example to highlight our approach's key features in a manageable fashion. Generally, we do not prescribe a dedicated concrete syntax for instances of the metamodel as requirements differ between different application domains and organizational environments. Figure 6 depicts a screenshot of the itemis YAKINDU Security Analyst. The Security Analyst provides different concrete syntaxes to work on instances of the metamodel. A textual concrete syntax for threats is displayed in the upper half of the screenshot (titled attack step in the syntax). The second half demonstrates a graphical concrete syntax of the SUD.

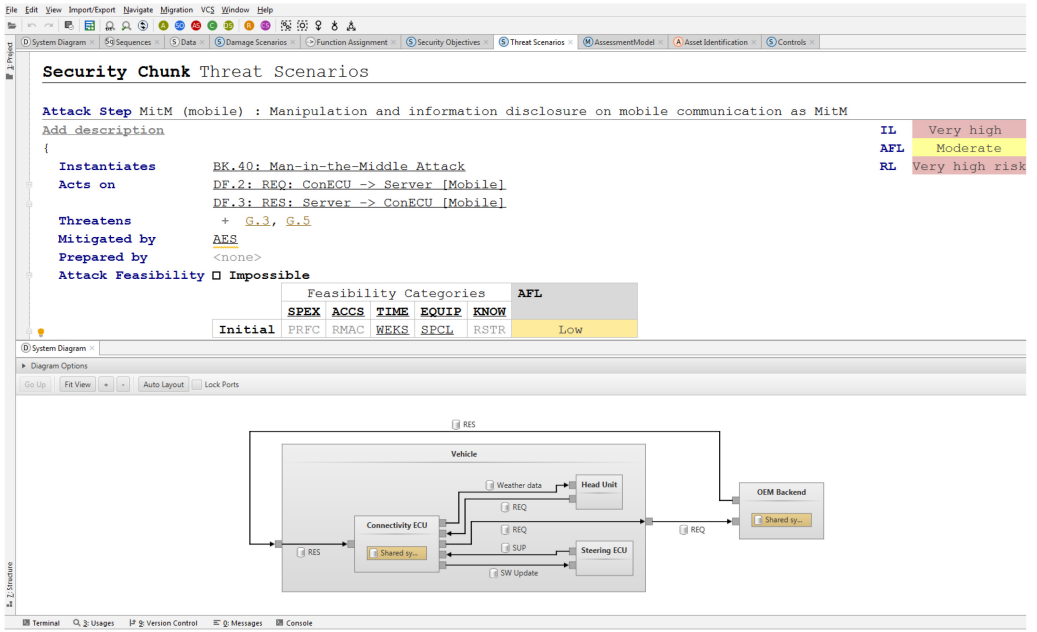

Fig. 6. A screenshot of concrete syntaxes provided by YAKINDU Security Analyst 21.1.

The example describes a fictitious software update function in which a server pushes an update into a vehicle. Violation of the security goal "Integrity of the update function" can lead to a safety-related damage scenario "Uncontrollable vehicle" as well as a damage scenario "Unauthorized tuning" related to financial losses. This security goal is threatened by two independent attack paths: the first attack path encompasses the combination of the threat "Reverse Engineering" *and* the threat "Man-in-the-Middle attack (mobile)." The latter threat acts on the mobile dataflow between server and vehicle. The control "AES GCM" protects the confidentiality and the integrity of the transferred data and thus mitigates the **Man-in-the-Middle (MitM)** attack. However, the control also depends on the confidentiality of the data item "AES key." In our example, all vehicles share the same symmetric key. Therefore, the security goal "Confidentiality of AES key" is threatened by a key extraction attack on a single vehicle.

The second attack path complements the **Man-in-the-Middle** (**MitM**) attack on the dataflow between server and vehicle with an attack on a dataflow inside the vehicle but still requires reverse engineering by the attacker. The control "AES GCM" does not protect dataflows inside the vehicle, as it only acts on the dataflow between server and vehicle. In this example, we also limit the attacker model to tuning-related attacks when physical access is needed. Consequently, the assumption "Tuning only" transforms the damage scenario "Uncontrollable vehicle" into the damage scenario "Unauthorized tuning." Note that damage transformation can also remove a damage scenario completely.

5 PROPAGATION RULES AND RISK CALCULATION

In the previous sections, we defined the metamodel and provided an example for an instance of the graph. In this section, we provide rules to calculate risk levels for a specific graph. First, we give an intuition of the idea. Then we formalize the actual calculation based on the metamodel elements instantiated in a graph. We conclude this section with an example.

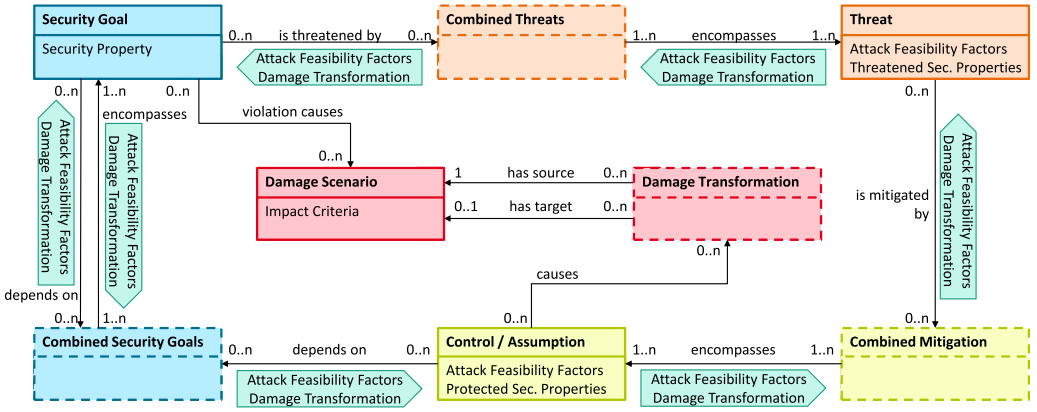

Fig. 7. The propagation of attack feasibility factors and damage transformation through the graph.

5.1 Intuition

In contrast to other **SRA** models, we aim to calculate the risk level for security goals, threats, controls, and damage scenarios individually. It is desirable to identify risk levels for all these elements, as identifying the most critical threats, the security goals and assets at highest risk, the weakest links among the controls, or the most critical damage scenarios all represent valuable information in making risk treatment decisions.

Calculation of risks requires two inputs: an estimation of the attack feasibility rating (based on attack feasibility factors) and an impact rating. These inputs are defined as attributes in separate metamodel elements of a graph instance: Threats and controls entail attack feasibility factor attributes. Damage scenarios entail impact criteria as attributes. Impact criteria attributes, in turn, are mapped to impact ratings in the assessment model. This results in specific impact ratings being available in damage scenarios. Furthermore, controls and assumptions may cause a damage transformation selecting relevant damage scenarios. We use the relations between the risk assessment-specific elements in the graph to combine these attributes and calculate risk levels. The combination of attributes brings together the two required inputs for risk calculation. Note that the metamodel technically allows for the definition of circular dependencies, but for the presented approach, the metamodel instance must be a DAG. In our practical experience on real life projects, this does not impose relevant limitations on the modeling capabilities.

The basic idea is to let the values of attributes flow or propagate through the graph. Figure 7 shows the propagation of attack feasibility factors and damage transformations. Figure 8 shows calculated risk values propagating in the opposite direction along the edges. The sequence of nodes from a control, assumption, or threat to another node creates an attack path toward that node. Note that we call every such path of any length an attack path. We calculate risks for every attack path toward a security goal. Any of the following types of nodes can be on an attack path and receive as well as propagate values: Security Goal, Combined Security Goals, Assumption, Control, Combined Mitigation, Threat, Combined Threats. The metamodel elements not listed (Damage Scenario and Damage Transformation) are used to calculate risk values but are not part of attack paths themselves.

Multiple attack paths can lead to a damage scenario (effectively forming an attack tree as part of the graph). The propagation rules define how to accumulate the values of attributes along the attack paths and how to combine multiple attack paths with each other. Consequently, changing an attribute at a node causes the risk values of all related nodes in the graph to be updated accordingly.

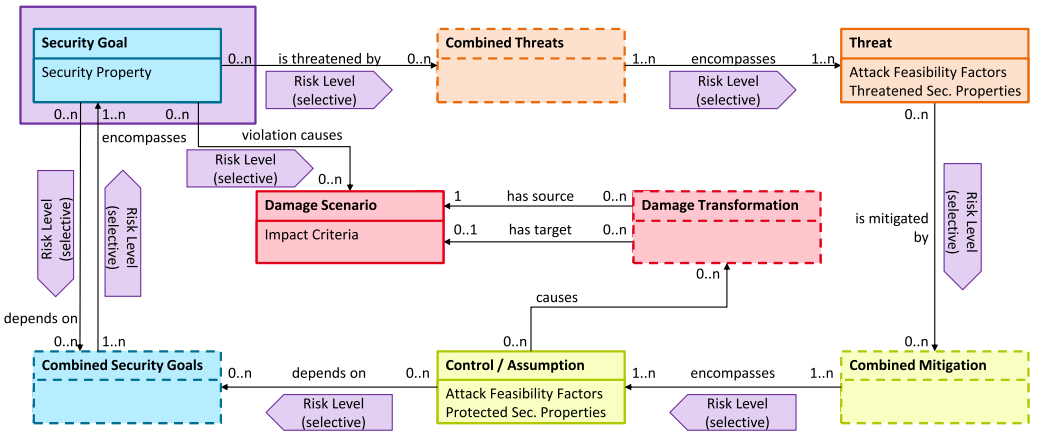

Fig. 8. The risks are propagated selectively, following the origins of the attack paths.

As noted above, the algorithm has two parts. In the first part (attack feasibility factor and damage transformation propagation), we create all attack paths of maximum length. This step starts in those assumptions, controls, and threats without any "depends on" or "is mitigated by" relations (these nodes are referred to as "leafs"). The algorithm first applies the calculation rules as defined below to all of the "leaf" nodes. As a result of these rules, a set of attack paths, all including the current node, is created as output and passed along the edge to the next node as input. Every node applies the propagation rules whenever all input sets (for all incoming edges) are available. Eventually, all nodes in the graph that can be part of an attack path are covered and the first part of the algorithm is finished.

The second part (risk propagation) works on the graph in the opposite direction. The algorithm starts in security goals, calculates risks for all incoming attack paths (propagated in step 1), and then propagates risks along the incoming attack paths of these security goals. Security goals possess relations to damage scenarios as a basis for the impact rating. However, the incoming attack paths might have transformation effects on these scenarios. After the effects are applied, a risk level is calculated for each attack path. The highest risk determines the risk level for the security goal itself. The risk level for each attack path subsequently travels along the attack path through the graph.

This modeling and calculation approach therefore enables risk decisions that include very complex dependencies. The following calculation instructions allow for an automated implementation.

5.2 Calculation

We split the description of calculation rules into two parts. First, we start with the propagation of attack feasibility factors and damage transformations as shown in Figure 7 with the rules presented in Table 1. Then we provide the propagation of risk levels, which flow in the opposite direction along the attack paths as shown in Figure 8 with the rules presented in Tables 2 and 3. The practical calculation also happens in this order.

5.2.1 Propagation of Attack Feasibility Factors and Damage Transformations. The first type of propagation concerns attack feasibility factors. As described in Section 4, they help to estimate the attack feasibility rating. The effort required by an attacker is at least the effort required to attack the initial node on the path, but usually increases along the path as different steps have to be taken, requiring more effort. Similarly, a node that causes damage transformation propagates

Table 1. Propagation Rules for Attack Feasibility Factors and Damage Transformation Effects

Threat		
Input	Sources: m connected **Combined Mitigations**	
	Value: $A := \bigcup_{i=1..m} A_i$ (the union of the incoming attack paths). The m connected **Combined Mitigation** nodes propagate m attack path sets A_i. These sets contain a total of $n = \sum_{i=1..m} \lvert A_i \rvert$ attack paths $p_j = (r_j, D_j, N_j)$.	
Local	r_0 := tuple of own attack feasibility factor values	e_0 := the node's unique ID
Output	Targets: All connected **Combined Threats** nodes	
	Value for $n = 0$ (a "leaf" node): $\{(r_0, \emptyset, \{e_0\})\}$, i.e., one attack path with the threat's own values	
	Value for $n > 0$: $\bigcup_{p_j \in A}\{\text{cpths}[(r_0, \emptyset, \{e_0\}), p_j]\} = \bigcup_{j=1..n}\{\text{cpths}[(r_0, \emptyset, \{e_0\}), (r_j, D_j, N_j)]\} =$ $\bigcup_{j=1..n}\{(\text{affmax}[r_0, r_j], \emptyset \cup D_j, \{e_0\} \cup N_j)\}$, i.e., the threat propagates n attack paths, where each of the n attack paths in the input set A is combined with the threat's attack feasibility factor values r_0 and node ID e_0.	
Combined Threats		
Input	Sources: m connected **Threats**	
	Value: m output sets A_i of the m connected **Threats**, each with $\lvert A_i \rvert$ attack paths for a total of $n = \sum_{i=1..m} \lvert A_i \rvert$ attack paths	
Local	$r_0 := \hat{r}$ (no own attack feasibility factor values)	e_0:= the node's unique ID
Output	Targets: All connected **Security Goal** nodes	
	Value: $A := \{\text{cpths}[p_1, \ldots, p_m, (r_0, \emptyset, \{e_0\})] \mid p_i \in A_i \text{ for every } i \in \{1, \ldots, m\}\}$. The output A contains $x := \prod_{i=1..m} \lvert A_i \rvert$ attack paths: It encompasses all possible combinations of incoming attack paths for each of m connected threats. Every threat contributes $\lvert A_i \rvert$ different attack paths. By combining all choices of selecting a single attack path for each threat, x different attack paths (with combined attack feasibility factor values) are created for the output set. The node's unique ID e_0 is added to each attack path.	
Security Goal		
Input	Sources: k connected **Combined Threats** nodes and m connected **Combined Security Goals** nodes	
	Value: $A := \bigcup_{i=1..k} A_i \cup \bigcup_{j=1..m} \hat{A}_j$ (the union of the incoming attack paths). The k connected **Combined Threats** nodes propagate k attack path sets A_i. Additionally, the m connected **Combined Security Goals** nodes propagate m attack path sets $\hat{A}_j$, resulting in a total of $n = \sum_{i=1..k} \lvert A_i \rvert + \sum_{j=1..m} \lvert \hat{A}_j \rvert$ incoming attack paths.	
Local	$r_0 := \hat{r}$ (no own attack feasibility factor values)	e_0 := the node's unique ID

(Continued)

Table 1. Continued

Output	Targets: All connected **Combined Security Goals** nodes	
	Value for $n = 0 : \emptyset$ (as the node is not attacked, nothing is propagated)	
	Value for $n > 0$: $\bigcup_{p_j \in A}\{\text{cpths}[p_j, (r_0, \emptyset, \{e_0\})]\}$, i.e., the node adds its unique ID e_0 to each attack path but does not influence the attack feasibility factors (as r_0 is set to minimum for each value) or the damage transformation effects.	
Combined Security Goals		
Input	Sources: m connected **Security Goals**	
	Value: m attack path sets A_i of the m connected **Security Goals**, each with $\lvert A_i \rvert$ attack paths for a total of $n = \sum_{i=1..m} \lvert A_i \rvert$ attack paths	
Local	$r_0 := \hat{r}$ (no own attack feasibility factor values)	e_0:= the node's unique ID
Output	Targets: All connected **Security Goal** and **Control** nodes	
	Value: $A := \{\text{cpths}[p_1, \ldots, p_m, (r_0, \emptyset, \{e_0\})] \mid p_i \in A_i \text{ for every } i \in \{1, \ldots, m\}\}$. The output A contains $x := \prod_{i=1..m} \lvert A_i \rvert$ attack paths: it encompasses all possible combinations of incoming attack paths for each of m connected security goals. Every security goal contributes $\lvert A_i \rvert$ different attack paths. By combining all choices of selecting a single attack path for each security goal, x different attack paths (with combined attack feasibility factor values) are created for the output set. The node's unique ID e_0 is added to each attack path.	
Control		
Input	Sources: m connected **Combined Security Goals** nodes	
	Value: $A := \bigcup_{i=1..m} A_i$ (the union of the incoming attack paths). The m connected **Combined Security Goals** nodes propagate m attack path sets A_i. These sets contain a total of $n = \sum_{i=1..m} \lvert A_i \rvert$ attack paths p_j.	
Local	r_0 := tuple of own attack feasibility factor values	
	D_0 := set of damage transformation effects	e_0 := the node's unique ID
Output	Targets: All connected **Combined Mitigation** nodes	
	Value for $n = 0$ (a "leaf node"): · if $D_0 = \emptyset$: $\{(r_0, \emptyset, \{e_0\})\}$, · else: $\{(r_0, \emptyset, \{e_0\}), (\hat{r}, D_0, \{e_0\})\}$, i.e., two attack paths for a control with at least one damage transformation effect or one attack path for a control without. Note that $\hat{r}$ represents the tuple of minimal attack feasibility factor values.	
	Value for $n > 0$: · if $D_0 = \emptyset$: $\bigcup_{p_j \in A}\{\text{cpths}[(\hat{r}, \emptyset, \{e_0\}), p_j]\} \cup \{(r_0, \emptyset, \{e_0\})\}$ · else: $\bigcup_{p_j \in A}\{\text{cpths}[(\hat{r}, \emptyset, \{e_0\}), p_j]\} \cup \{(r_0, \emptyset, \{e_0\}), (\hat{r}, D_0, \{e_0\})\}$	

(Continued)

Table 1. Continued

	i.e., a control without damage transformation effects propagates $n + 1$ attack paths. This includes the n attack paths in the input set A combined with the control's node ID e_0. As the control is broken via its dependencies, the control's attack feasibility factor value tuple r_0 is not added to these paths. The $(n + 1)$th propagated attack path is the same as for a "leaf" node, as the control is not broken via its dependencies in this case. A control with damage transformation effects propagates one additional $(n + 2)$th attack path without the node's attack feasibility factor values, but its damage transformation effects D_0 and node ID e_0. This reflects an attacker's option to accept the control's effects on damages instead of breaking the control.
Assumption	
Input	Sources: None (assumptions are always "leaf" nodes)
	Value: $A := \emptyset$ (no incoming attack paths).
Local	D_0 := set of damage transformation effects | e_0 := the node's unique node ID
	r_0 := tuple of own attack feasibility factor values OR $\perp$ (where $\perp$ means that the assumption always causes a damage transformation effect)
Output	Targets: All connected **Combined Mitigation** nodes
	Value: · if $r_0 = \perp$ and $D_0 = \emptyset$: $\emptyset$, · if $r_0 = \perp$ and $D_0 \neq \emptyset$: $\{(\hat{r}, D_0, \{e_0\})\}$, · if $r_0 \neq \perp$ and $D_0 = \emptyset$: $\{(r_0, \emptyset, \{e_0\})\}$, · if $r_0 \neq \perp$ and $D_0 \neq \emptyset$: $\{(\hat{r}, D_0, \{e_0\}), (r_0, \emptyset, \{e_0\})\}$, i.e., no attack paths for an assumption without effects, one attack path for an assumption with only one effect, and two attack paths for an assumption with attack feasibility factor values and a damage transformation effect.
Combined Mitigations	
Input	Sources: m connected **Control** or **Assumption** nodes
	Value: m output sets A_i of the m connected nodes (**Controls** / **Assumptions**), each with $\|A_i\|$ attack paths for a total of $n = \sum_{i=1..m} \|A_i\|$ attack paths
Local	$r_0 := \hat{r}$ (no own attack feasibility factor values) | e_0:= the node's unique ID;
Output	Targets: All connected **Threat** nodes
	Value: $A := \{\text{cpths}[p_1, \ldots, p_m, (r_0, \emptyset, \{e_0\})] \mid p_i \in A_i \text{ for every } i \in \{1, \ldots, m\}\}$. The output A contains $x := \prod_{i=1..m} \|A_i\|$ attack paths: It encompasses all possible combinations of incoming attack paths for each of m connected mitigations (controls / assumptions). Every mitigation contributes $\|A_i\|$ different attack paths. By combining all choices of selecting a single attack path for each mitigation, x different attack paths (with combined attack feasibility factor values) are created for the output set. The node's unique ID e_0 is added to each attack path.

this effect along the attack path. The propagation starts in assumptions, as well as in controls and threats without any "depends on" or "is mitigated by" relations. Nodes with the term "Combined" in their node type combine the incoming attack paths as defined below. Threats affect incoming attack paths by adding their own attack feasibility factors to the attack path. Mitigations (controls

Table 2. Propagation Rules for Risk Results of **Security Goal** Nodes

Security Goal		
Input	Sources: All o **Security Goals** where this node is on an attack path (see *Output* in this table)	
	Value: $U := \bigcup_{i=1..o} U_i$ (the union of the incoming results). The o **Security Goals** propagate o result sets U_i. These sets contain a total of $q = \sum_{i=1..o} \lvert U_i \rvert$ results $u_i = (\rho_i, N_i)$.	
Input	$A :=$ the set of n incoming attack paths (r_i, D_i, N_i) for $i = 1..n$ (see Table 1)	
Local	$S_0 :=$ the set of connected **Damage Scenario** nodes	$e_0 :=$ the node's unique ID
Derived	$S_{i,0} := \bigcup_{d \in D_i, s \in S_0} \{\mathrm{dt}[d, s]\}$, the set of transformed damage scenarios for attack path i after one iteration	
	$S_{i,k} := \bigcup_{d \in D_i, s \in S_{i,k-1}} \{\mathrm{dt}[d, s]\}$, the set of transformed damage scenarios after $k + 1$ iterations	
	$S_i := S_{i,x}$ with $S_{i,x} = S_{i,x-1}$, the resulting set of transformed damage scenarios. Note that this allows cycles or ambiguous situations. It is up to the analyst creating the model to prevent or resolve such issues. $S_i = S_0$ if $D_i = \emptyset$.	
	$\eta_i := \mathrm{imr}[\bigcup_{s_j \in S_i} \mathrm{dc}[s_j]]$, the impact rating for attack path i	
	$\alpha_i := \mathrm{afr}[r_i]$, the attack feasibility rating for attack path i	
	$\rho_i := \mathrm{rl}[\alpha_i, \eta_i]$, the risk level for attack path i	
	$R := \bigcup_{i=1..\lvert A \rvert} \{(\rho_i, N_i)\}$, the set of all risk level results for all incoming attack paths in A	
	$\rho := \max[\{\rho_i \mid (\rho_i, N_i) \in R\} \cup \{\rho_j \mid (\rho_j, N_j) \in U\}]$, the security goal's risk value, i.e., the maximum of the node's own risk levels and the risk levels propagated to the node	
Output	Targets: All nodes e_i on the incoming attack paths with $e_i \in N_j$ and $(\rho_j, N_j) \in R$	
	Value: $R_{e_i} := \{(\rho_j, N_j \cup \{e_0\}) \mid (\rho_j, N_j) \in R \text{ and } e_i \in N_j\}$. Propagate to the target node e_i the results R_{e_i} for all attack paths (local and propagated) with the security goal's ID e_0 on the path. Note that U might be empty (for a "leaf" node with $q = 0$).	
Output	Targets: $\lvert S_i \rvert$ connected **Damage Scenario** nodes $s \in S_i$	
	Value: $R'_i := \{(\mathrm{rl}[\alpha_i, \mathrm{imr}[\mathrm{dc}[s]]], N_i) \mid \alpha_i = \mathrm{afr}[r_i] \text{ for all } i \in \{1, \ldots, n\} \text{ and } s \in S_i\}$. Propagate to each connected damage scenario $s \in S_i$ the result set R'_i, containing a risk value and the nodes on the attack path N_i. The risk value is calculated from the attack feasibility rating α_i for that attack path and the damage rating $\mathrm{imr}[\mathrm{dc}[s]]$ for damage scenario s. Note that S_i contains the damage scenarios for an attack path *after* damage transformation.	

or assumptions) always generate one or more new attack paths: The first attack path breaks the mitigation via its attack feasibility factors. In this case, the mitigation's attack feasibility factors are propagated in a new attack path. The second attack path is generated if the mitigation has damage transformation effects. This attack path leaves the mitigation in place and propagates the damage transformation effect. If the mitigation has incoming attack paths, then these paths break the mitigation via its dependencies. Consequently, these attack paths are propagated without

Table 3. Propagation Rules for Risk Results of **Combined Security Goals**, **Combined Threats**, **Threat**, **Combined Mitigation**, **Control** or **Assumption** Nodes

Combined Security Goals, **Combined Threats**, **Threat**, **Combined Mitigation**, **Control** or **Assumption**		
Input	Sources: All o **Security Goals** where this node is on an attack path (see Table 2)	
	Value: $U := \bigcup_{i=1..o} U_i$ (the union of the incoming results). The o **Security Goals** propagate o result sets U_i. These sets contain a total of $q = \sum_{i=1..o} \lvert U_i \rvert$ results $u_i = (\rho_i, N_i)$.	
Local	e_0 := the node's unique ID	
Derived	$\rho := \max[\{\rho_i \mid (\rho_i, N_i) \in U\}]$, the node's risk value, i.e., the maximum of the risk levels propagated to the node	
Output	Targets: None	Value: None

changes to the attack feasibility factors or damage transformation effects. Security goals always propagate attack paths without changes to the attack feasibility factors or damage transformation effects. Attack paths usually terminate in security goals, but may also terminate in other nodes (e.g., when a threat does not violate any security goals).

Table 1 defines the propagation rules for attack feasibility factors and damage transformation effects. We use the following notation and definitions:

Let r denote a tuple of attack feasibility factor values used to determine the attack feasibility rating. For example, using an approach based on attack potentials (cf. Reference [6]): Given an assessment model with f attack feasibility factors, let $r_i := (v_{i,1}, \ldots, v_{i,f})$ denote an f-tuple of attack feasibility factor values, where $v_{i,n}$ represents the value of attack feasibility factor n for the attack feasibility factor tuple r_i.

$\hat{r}$ represents minimum values for each attack feasibility factor. For example, using an approach based on attack potentials (cf. Reference [6]), this results in the tuple $(0, \ldots, 0)$.

Every node in the graph has a unique ID, e_i. The ID 0 is reserved for "no node." We use the "no node" concept for a damage transformation that completely removes a damage scenario. E represents the set of all node IDs. $S \subset E$ represents the set of all *Damage Scenario* node IDs (including 0), and $T \subset E$ represents the set of all *Damage Transformation* node IDs.

A Damage Transformation node d has a relation "has source" to exactly one Damage Scenario node $s \in S$ and another relation "has target" to another Damage Scenario node $t \in S$. Let $\text{src} : T \to S$ return a damage transformation's source node s and $\text{tgt} : T \to S$ return its target node t. The damage transformation function $\text{dt} : (T, S) \to S$ uses a Damage Transformation node $d \in T$ and a Damage Scenario node $s \in S$ as input and provides a Damage Scenario node $s' \in S$ as output. It is defined as

$$\text{dt}[d, s] := \begin{cases} s & s \neq \text{src}[d] \\ \text{tgt}[d] & s = \text{src}[d] \end{cases}.$$

Let $p := (r, D, N)$ define an *attack path*, where $D \subseteq T$ represents a set of damage transformation nodes and N represents the set of nodes with IDs e_i traversed on the attack path. In other words, p defines the effects on risk accumulated in a single attack path toward a node in the graph and combines an attack effort (attack feasibility factor values in r) with zero or more damage transformation effects in D. Note that, given the acyclic nature of the graph, storing the IDs in a set is sufficient to reconstruct a full attack path from a given starting point. Note that several attack paths may contain the same set of nodes, as, e.g., controls with damage transformation effects create two attack paths.

Let R denote the set of all attack feasibility factor value tuples r. Then affmax : $R^* \rightarrow R$ denotes a function that takes an arbitrary number $k \in \mathbb{N}$ of attack feasibility factor value tuples $r_1, \ldots, r_k$ as input and calculates the maximum for each attack feasibility factor. For example, using an approach based on attack potentials (cf. Reference [6]) as attack feasibility factors, we obtain

$$\text{affmax}[r_1, \ldots, r_k] := (\max[v_{1,1}, \ldots, v_{k,1}], \ldots, \max[v_{1,f}, \ldots, v_{k,f}]).$$

Let P denote the set of all attack paths p. Then cpths : $P^* \rightarrow P$ denotes a function that takes an arbitrary number $k \in \mathbb{N}$ of attack paths $p_1, \ldots, p_k$ as input and calculates the maximum value for each attack feasibility factor, while damage transformation effects are unaffected and all nodes on the path are remembered. More precisely,

$$\text{cpths}[p_1, \ldots, p_k] := \left(\text{affmax}[r_1, \ldots, r_k], \bigcup_{j=1..k} D_j, \bigcup_{j=1..k} N_j \right).$$

The output of cpths[] is itself an attack path that accumulates the values and effects of all the inputs.

Finally, let A denote a set of propagated attack paths. For each node, the set of all incoming attack paths represents the input set of the calculation step. This input set is then combined with the node's own values to propagate a number of attack paths along the graph in the node's output set.

Table 1 provides specifics on the calculation and propagation rules. Note that variables are redefined for each node and node type (e.g., a node's unique ID is always e_0 in the node's scope). Names of metamodel elements are in bold.

5.2.2 Risk Propagation. With the attack feasibility factor values and the damage transformation effects completely propagated, the calculation and propagation of risk levels takes place. The initial calculation happens in Security Goal nodes without an incoming "encompasses" relation of a Combined Security Goals node (i.e., a "leaf" node for risk propagation). For each attack path, we calculate the attack feasibility rating based on the attack path's attack feasibility factor values. Next, we apply all damage transformation effects of the selected attack path to the Security Goal node's related Damage Scenario nodes (following the relation "violation causes"). The resulting damage scenarios provide impact criteria, which in turn provide impact ratings. The highest of these impact ratings defines the impact rating for the selected attack path. Combining this impact rating with the attack feasibility rating of the attack path provides a risk level for the attack path. The highest risk value of any of a node's attack paths defines the risk for the node itself. The node then propagates all risk levels along its respective attack paths to propagate the results through the graph. We use the following additional notation for these rules:

Let $\alpha := \text{afr}[r]$ calculate the attack feasibility rating for a attack feasibility factor value tuple r. Note that this function is part of the assessment model and thus not defined here. Let $\text{dc}[s] := C$ return the set C of impact criteria assigned to a damage scenario s. Let $\eta := \text{imr}[C]$ calculate the impact rating for a set of impact criteria C, based on the assessment model. For example, imr[] might return the maximum over a set of impact ratings assigned to each impact criterion in the assessment model (where higher numbers imply higher impacts). Let $\rho := \text{rl}[\alpha, \eta]$ calculate the risk level ρ for an attack feasibility rating α and an impact rating η.

Let $u := (\rho, N)$ represent a risk level result for an attack path with risk level ρ and a set of node IDs N. Finally, let U denote a set of risk level results.

Tables 2 and 3 define the specifics for the risk level calculation inside a Security Goal node and the risk level result propagation. Table 2 also describes the risk level propagation to Damage

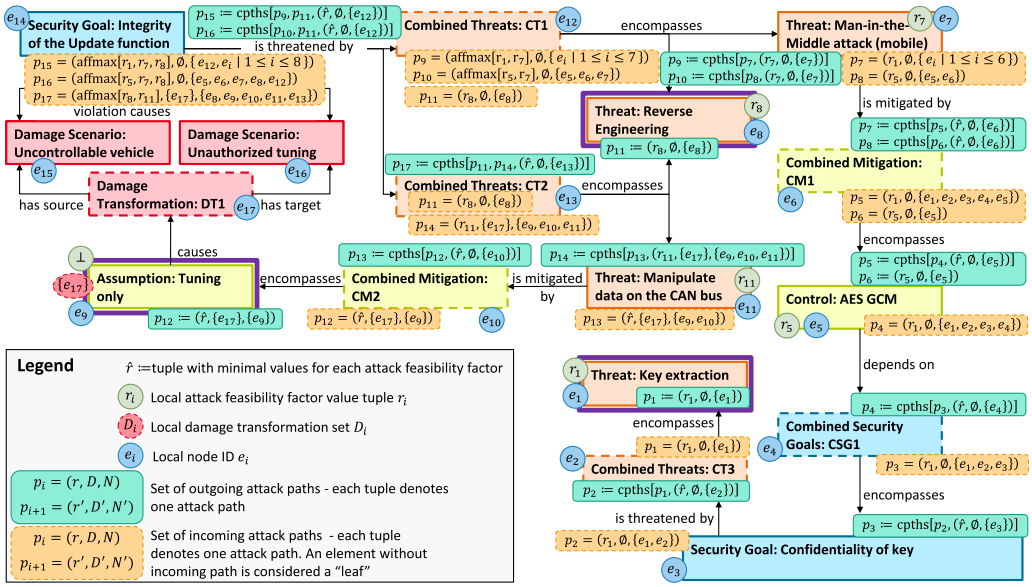

Fig. 9. Propagation of attack factors and damage transformation shown on the software update example. Thick borders mark "leaf" nodes.

Scenario nodes (see second "Output" in the table). Note that instead of propagating risks hop by hop on an attack path, we propagate selected risks directly from Security Goal nodes. The derived risk level result for each attack path is propagated to each node on this attack path. This simplifies the propagation rules, while preserving the semantics of Figure 8.

5.3 Example

We use our example presented in Section 4.5 to depict the propagation of attack feasibility factors and damage transformation effects within that example in Figure 9. Thick borders mark the three nodes considered as "leaf" nodes for this propagation. These leafs serve as starting points for the propagation. Note that we use unique identifiers for the propagated attack paths in this example and not the indices enumerating elements in a set of attack paths. Rounded boxes with solid borders denote sets of outgoing attack paths, while rounded boxes with dashed borders denote sets of incoming attack paths. Local attributes are depicted as circles.

This results in three attack paths for the security goal "Integrity of the Update function."
(1) One way to attack this security goal is to extract the private key, conduct reverse engineering, and manipulate the encrypted and signed software update as man-in-the-middle on the mobile connections. The threat "Key extraction" has its attack feasibility factor values r_1. These are propagated through the security goal "Confidentiality of key" to the control "AES GCM" that depends on this security goal. This results in a broken control and consequently does not add the control's attack feasibility factor values to the attack path. The attack path is then propagated to the threat "Man-in-the-Middle attack (mobile)." This threat has its attack feasibility factor values r_7. affmax$[r_1, r_7]$ calculates the maximum value for each attack feasibility factor, while afr[affmax$[r_1, r_7]$] calculates the attack feasibility rating. To threaten the target security goal "Integrity of the Update function," the threat "Reverse Engineering" with its attack feasibility factors r_8 needs to be combined with the MitM threat in node "CT1" and thus, a total attack feasibility

rating of afr[affmax[r_1, r_7, r_8]] is calculated for this attack path. The traversed node IDs are accumulated along the path, while no damage transformation effects are encountered.
(2) A second attack is to break the control "AES GCM" with its attack feasibility factor values r_5 (not by extracting the key, but e.g., by brute-forcing it because of a short key length), conduct reverse engineering, and manipulate the encrypted and signed software update as man-in-the-middle on the mobile connections. This results in a total attack feasibility rating of afr[affmax[r_5, r_7, r_8]], no damage transformation effects, and a different set of node IDs compared to attack path 1.
(3) A third attack is to conduct reverse engineering and manipulate the software update on the CAN bus inside the vehicle, where no encryption is applied. For this attack, the assumption "Tuning only" causes damage transformation node "DT1" with ID e_{17}, transforming "Uncontrollable vehicle" to "Unauthorized tuning." This is propagated to the threat "Manipulate data on the CAN bus," which possesses the attack feasibility factor values r_{11}. Together with "Reverse Engineering" this combines to the total of afr[affmax[r_8, r_{11}]]. The damage transformation set for this attack path is $\{e_{17}\}$.

A risk level is calculated for each of the three attack paths, based on each attack path's attack feasibility rating as well as the damage associated with each damage scenario after damage transformation. For our three attack paths, this results in

- rl[afr[affmax[r_1, r_7, r_8]], imr[dc[e_{15}], dc[e_{16}]]] (attack path 1)
- rl[afr[affmax[r_5, r_7, r_8]], imr[dc[e_{15}], dc[e_{16}]]] (attack path 2)
- rl[afr[affmax[r_8, r_{11}]], imr[dc[e_{16}]]] (attack path 3).

Furthermore, each of these risks is propagated to all nodes on the respective attack path. The highest of these risks determines the risk level for each node.

Finally, a risk level is calculated for each attack path combined with the resulting damage scenarios after damage transformation on that attack path, i.e.,

- rl[afr[affmax[r_1, r_7, r_8]], imr[dc[e_{15}]]] (attack path 1, damage scenario with ID e_{15})
- rl[afr[affmax[r_1, r_7, r_8]], imr[dc[e_{16}]]] (attack path 1, damage scenario with ID e_{16})
- rl[afr[affmax[r_5, r_7, r_8]], imr[dc[e_{15}]]] (attack path 2, damage scenario with ID e_{15})
- rl[afr[affmax[r_5, r_7, r_8]], imr[dc[e_{16}]]] (attack path 2, damage scenario with ID e_{16})
- rl[afr[affmax[r_8, r_{11}]], imr[dc[e_{16}]]] (attack path 3, damage scenario with ID e_{16}).

6 CONCLUSION

Security risk analyses are becoming a mandatory development step in many domains due to international regulations. This is already the case in the automotive domain due to the new UN Regulation No. 155 [36]. Implementing the necessary processes for systematically evaluating complex systems, such as modern cars, is a demanding task. We address the question of how to implement such a process with a proven, model-based approach.

The structure of the presented model, in combination with the methodical approach, forms a common basis upon which system and security engineers can jointly develop and assess an instanced model. To this aim, we mix two artifact types, system and security properties with limited expressiveness, which proved well applicable in our experience.

Our metamodel encompasses the **SUD** itself, composed of functions, data elements, components, and dataflows. A fixed set of relations links these elements. We extend this metamodel to include the elements specific for **SRA**s (security goals, threats, controls, and assumptions) and additional relations. Thus, we achieve an integrated representation of the **SUD** as well as its security properties in the context of an **SRA**. To properly consider the often-intricate dependencies and influences, we introduce a set of propagation rules. Consequently, the relations between the security-specific

elements can be validated by tracing them back to the elements of the **SUD**. A consequence of this mixed artifacts strategy is that security modeling requires repeating or extending several steps of functional modeling, i.e., creating use case diagrams and flowcharts, which is not an easily automatable task.

In contrast to other approaches that separate elements and relations of the **SUD** and the security-specific elements, we provide an integrated perspective that allows users to assess the level of risk and the impact of threats, controls, and assumptions in a qualified manner. Local or iterative changes to the model rarely require changes to other elements due to the modular structure. This supports the maintainability of analyses, reduces the follow-ups' efforts on updates or new findings, and improves comprehensibility.

To demonstrate our method, we present an application on small fictitious example. In the absence of suitable evaluation criteria for the quality of analyses, it is not yet possible to measure the quality of the applied approach beyond that. The development of such evaluation criteria is the subject of ongoing research. For this purpose, we intend to re-analyze existing assessments to identify relevant properties. However, we collected evidence of our method's suitability in several hundred real-life security risk assessments in projects with industrial customers. We conducted security risk assessments for the development of vehicle functions and ECUs, industrial components, IT systems, and IoT devices in the course of 10 years and continuously improved the method based on our own experience and the feedback of our customers.

Limitations of our approach include an increase in the complexity of the resulting models, requiring security experts to apply the method. In the future, expressive and case-specific **SRA**s will no longer be sufficient. It will be required to infer between different models and evaluate them simultaneously to cope with complex, integrated systems. This will require additional methods. Likewise, expertise is required to tailor the assessment model and catalogs to a company's needs for best results. Additionally, achieving higher precision often comes at the price of increasing model complexity.

Creation and maintenance of these models gain from tool support, such as the Yakindu Security Analyst.

Although initially developed for automotive security risk analysis, we successfully applied the proposed structure and representation as graphs in other domains, such as industrial security.

REFERENCES

[1] Daniel Angermeier, Kristian Beilke, Gerhard Hansch, and Jörn Eichler. 2019. Modeling security risk assessments. In *Proceedings of the 17th Embedded Security in Cars (ESCAR Europe'19)*). Ruhr-Universität Bochum, Bochum, Germany, 133–146. https://doi.org/10.13154/294-6670

[2] Daniel Angermeier, Alexander Nieding, and Jörn Eichler. 2016. Supporting risk assessment with the systematic identification, merging, and validation of security goals. In *International Workshop on Risk Assessment and Risk-driven Testing*. Springer, Cham, Germany, 82–95.

[3] Daniel Angermeier, Alexander Nieding, and Jörn Eichler. 2016. Systematic identification of security goals and threats in risk assessment. *Softwaretechnik-Trends* 36, 3 (2016).

[4] Daniel Angermeier, Alexander Nieding, and Jörn Eichler. 2017. Supporting risk assessment with the systematic identification, merging, and validation of security goals. In *Risk Assessment and Risk-Driven Quality Assurance*, Jürgen Großmann, Michael Felderer, and Fredrik Seehusen (Eds.). Springer International Publishing, Cham, 82–95.

[5] George E. P. Box. 1979. Robustness in the strategy of scientific model building. In *Robustness in Statistics*, Robert L. Launer and Graham N. Wilkinson (Eds.). Elsevier, Madison, WI, 201–236. https://doi.org/10.1016/B978-0-12-438150-6.50018-2

[6] Common Criteria Editorial Board. 2017. *Common Methodology for Information Technology Security Evaluation: Evaluation Methodology* (3.1r5 ed.). Standard. Common Criteria.

[7] J. Eichler. 2015. *Model-based Security Engineering for Electronic Business Processes*. Ph. D. Dissertation. Technische Universität München.

[8] Jörn Eichler and Daniel Angermeier. 2015. Modular risk assessment for the development of secure automotive systems. In *Proceedings of the VDI/VW-Gemeinschaftstagung Automotive Security*, Vol. 2263. VDI, Düsseldorf, 81–90.
[9] Benjamin Fabian, Seda Gürses, Maritta Heisel, Thomas Santen, and Holger Schmidt. 2010. A comparison of security requirements engineering methods. *Require. Eng.* 15, 1 (2010), 7–40. https://doi.org/10.1007/s00766-009-0092-x
[10] Shamal Faily, John Lyle, Cornelius Namiluko, Andrea Atzeni, and Cesare Cameroni. 2012. Model-driven architectural risk analysis using architectural and contextualised attack patterns. In *Proceedings of the Workshop on Model-Driven Security (MDsec'12)*. ACM, New York, NY, Article 3, 6 pages. https://doi.org/10.1145/2422498.2422501
[11] Jack Freund and Jack Jones. 2015. *Measuring and Managing Information Risk: A FAIR Approach*. Butterworth-Heinemann, Oxford, UK.
[12] Dimitris Gritzalis, Giulia Iseppi, Alexios Mylonas, and Vasilis Stavrou. 2018. Exiting the risk assessment maze: A meta-survey. *ACM Comput. Surv.* 51, 1, Article 11 (January 2018), 30 pages. https://doi.org/10.1145/3145905
[13] Mohammad Hamad and Vassilis Prevelakis. 2020. SAVTA: A hybrid vehicular threat model: Overview and case study. *Information* 11, 5 (May 2020), 273.
[14] Gerhard Hansch, Peter Schneider, and Gerd S. Brost. 2019. Deriving impact-driven security requirements and monitoring measures for industrial IoT. In *Proceedings of the 5th ACM Cyber-Physical System Security Workshop (CPSS'19)*. ACM, New York, NY, 37–45. https://doi.org/10.1145/3327961.3329528
[15] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt. 2015. P^2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language. *IEEE Trans. Depend. Sec. Comput.* 12, 6 (November 2015), 626–639. https://doi.org/10.1109/TDSC.2014.2382574
[16] IEC. 2020. *IEC 62443-3-2:2020 Security for Industrial Automation and Control Systems–Part 3-2: Security Risk Assessment for System design*. Standard. International Electrotechnical Commission and others, Geneva, CH.
[17] ISO/SAE. 2021. *ISO/SAE 21434:2021 Road Vehicles–Cybersecurity engineering*. Standard. International Organization for Standardization, Geneva, CH.
[18] Loren Kohnfelder and Praerit Garg. 1999. *The Threats to Our Products*. Technical Report. Microsoft Interface.
[19] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. 2014. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Comput. Sci. Rev.* 13 (November 2014), 1–38. https://doi.org/10.1016/j.cosrev.2014.07.001
[20] Michael Krisper, Jürgen Dobaj, Georg Macher, and Christoph Schmittner. 2019. RISKEE: A risk-tree based method for assessing risk in cyber security. In *Proceedings of the 26th European Conference on Systems, Software and Services Process Improvement (EuroSPI'19)*. Springer, Cham, Germany, 45–56. https://doi.org/10.1007/978-3-030-28005-5_4
[21] Katsiaryna Labunets, Fabio Massaci, and Alessandra Tedeschi. 2017. Graphical vs. Tabular notations for risk models: On the role of textual labels and complexity. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'17)*. IEEE Press, Los Alamitos, CA, 267–276. https://doi.org/10.1109/ESEM.2017.40
[22] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. 2010. *Model-Driven Risk Analysis: The CORAS Approach*. Springer Science & Business Media, Berlin, Germany. https://doi.org/10.1007/978-3-642-12323-8
[23] Feng Luo, Shuo Hou, Xuan Zhang, Zhenyu Yang, and Wenwen Pan. 2020. Security risk analysis approach for safety-critical systems of connected vehicles. *Electronics* 9, 8 (August 2020), 1242.
[24] Georg Macher, Harald Sporer, Reinhard Berlach, Eric Armengaud, and Christian Kreiner. 2015. SAHARA: A security-aware hazard and risk analysis method. In *Proceedings of the 2015 Design, Automation and Test in Europe Conference (DATE'15)*. 621–624.
[25] Charlie Miller and Chris Valasek. 2013. Adventures in automotive networks and control units. *Def. Con.* 21 (2013), 260–264.
[26] Jean-Philippe Monteuuis, Aymen Boudguiga, Jun Zhang, Houda Labiod, Alain Servel, and Pascal Urien. 2018. SARA: Security automotive risk analysis method. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*. Association for Computing Machinery, New York, NY, 3–14.
[27] Martin J. O'Connor and Amar K. Das. 2009. SQWRL: A query language for OWL. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions (OWLED'09, Vol. 529)*. CEUR-WS.org, Aachen, Germany, 208–215.
[28] N. Poolsappasit, R. Dewri, and I. Ray. 2012. Dynamic security risk management using bayesian attack graphs. *IEEE Trans. Depend. Sec. Comput.* 9, 1 (2012), 61–74. https://doi.org/10.1109/TDSC.2011.34
[29] Alastair Ruddle, Benjamin Weyl, Sajid Idrees, Y. Roudier, Michael Friedewald, Timo Leimbach, A. Fuchs, S. Gürgens, O. Henninger, Roland Rieke, M. Ritscher, H. Broberg, L. Apvrille, R. Pacalet, and Gabriel Pedroza. 2009. Security requirements for automotive on-board networks based on dark-side scenarios. *Seventh Research Framework Programme (2007–2013) of the European Community*. Deliverable D2.3: EVITA. E-safety vehicle intrusion protected applications, 150 pages. https://www.researchgate.net/profile/Gabriel-Pedroza/publication/304525166_Security_requirements_for_automotive_on-board_networks_based_on_dark-side_scenarios/links/57b06d4808ae15c76cba2666/Security-requirements-for-automotive-on-board-networks-based-on-dark-side-scenarios.pdf.

[30] Christoph Schmittner, Thomas Gruber, Peter Puschner, and Erwin Schoitsch. 2014. Security application of failure mode and effect analysis (FMEA). In *Computer Safety, Reliability, and Security*, Andrea Bondavalli and Felicita Di Giandomenico (Eds.). Springer International Publishing, Cham, 310–325.

[31] Adam Shostack. 2014. *Threat Modeling: Designing for Security*. John Wiley & Sons, Indianapolis, IN.

[32] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm. 2013. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Syst. J.* 7, 3 (December 2013), 363–373. https://doi.org/10.1109/JSYST.2012.2221853

[33] Amina Souag, Camille Salinesi, Raúl Mazo, and Isabelle Comyn-Wattiau. 2015. A security ontology for security requirements elicitation. In *Engineering Secure Software and Systems (ESSoS'15)*, Frank Piessens, Juan Caballero, and Nataliia Bielova (Eds.). Springer International Publishing, Cham, Germany, 157–177. https://doi.org/10.1007/978-3-319-15618-7_13

[34] UNECE WP.29 TF CS and OTA. 2020. *UN Regulation on Uniform Provisions Concerning the Approval of Vehicles with Regards to Cyber Security and Cyber Security Management System*. Proposal. UN World Forum for the Harmonization of Vehicle Regulations (WP.29).

[35] Jan Wolf, Felix Wieczorek, Frank Schiller, Gerhard Hansch, Norbert Wiedermann, and Martin Hutle. 2016. Adaptive modelling for security analysis of networked control systems. In *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR'16)*. BCS Learning & Development, Swindon, UK, 64–73. https://doi.org/10.14236/ewic/ICS2016.8

[36] UNECE GRVA WP29. 2021. *UN Regulation No. 155—Cyber Security and Cyber Security Management System*. Technical Report. United Nations.

[37] Peng Xie, Jason H. Li, Xinming Ou, Peng Liu, and Renato Levy. 2010. Using bayesian networks for cyber security analysis. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (DSN'10)*. IEEE Computer Society, Los Alamitos, CA, 211–220. https://doi.org/10.1109/DSN.2010.5544924

Received 28 June 2021; revised 27 May 2022; accepted 17 September 2022