Automating Security Risk and Requirements Management for Cyber-Physical Systems

Dissertation
zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
“Doctor rerum naturalium”
der Georg-August-Universität Göttingen

im Promotionsprogramm Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von
Gerhard Hansch

aus Regensburg
Göttingen, 2020

Betreuungsausschuss

Prof. Dr. Ramin Yahyapour
Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen mbH (GWDG),
Institut für Informatik, Georg-August-Universität Göttingen
Prof. Dr. Jens Grabowski
Institut für Informatik, Georg-August-Universität Göttingen

Mitglieder der Prüfungskommission

Referent: Prof. Dr. Ramin Yahyapour
Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen mbH (GWDG),
Institut für Informatik, Georg-August-Universität Göttingen
Korreferent: Prof. Dr. Jens Grabowski
Institut für Informatik, Georg-August-Universität Göttingen

Weitere Mitglieder der Prüfungskommission

Prof. Dr.-Ing. Marcus Baum
Institut für Informatik, Georg-August-Universität Göttingen
Prof. Dr. Dieter Hogrefe
Institut für Informatik, Georg-August-Universität Göttingen
Prof. Dr. Florin Manea
Institut für Informatik, Georg-August-Universität Göttingen
Prof. Dr.-Ing. Delphine Reinhardt
Institut für Informatik, Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 15. Oktober 2020

Hansch, Gerhard:
Automating Security Risk and Requirements Management for Cyber-Physical Systems

ORCID: 0000-0001-6906-6495

DOI: 10.24406/AISEC-N-608669

Acknowledgments

The work that led to this thesis was made possible by the support of many outstanding people I would like to acknowledge here.

First, I would like to thank my academic supervisors. Professor Dr. Ramin Yahyapour supported me with his excellent guidance, fruitful discussions, and patience. My second supervisor, Professor Dr. Jens Grabowski, gave me valuable feedback and asked the right questions during our meetings.

Next, I would like to thank the Fraunhofer-Institute for Applied and Integrated Security AISEC to enable and support this work. I am especially grateful for the support of my department head, Bartol Filipovic, and his deputy, Daniel Angermeier. They gave me the freedom to perform my research, learn about the industrial side of research, and attend international conferences. Daniel also continuously provided me with inspiration and critical feedback for the concepts in this work. Additionally, I would like to acknowledge the excellent cooperation with all my colleagues who have contributed in their particular way, from co-authoring my prior publications, over discussing profound technical aspects, up to bringing joy into everyday work.

My special gratitude goes to my colleagues Dr. Konstantin Böttinger, Peter Schneider, Dr. Sven Plaga, Dr. Johannes Obermaier, Alexander Kiening, Karla Markert, Stefan Tatschner, and Phillip Zieris. Your support has made this work possible. It is always an honor to work with you.

Additionally, I am grateful for the support of Christian Folda, Stephan Gerhager, Christoph Schmittner, Christian Staudte, Dr. Sven Bingert, Prof. Dr.-Ing. Jürgen Mottok, and Prof. Dr. Rudolf Hackenberg. Thank you for your support in finding the right track at the beginning of my journey.

My deepest gratitude belongs to my family and my closest friends for always supporting me, last but not least to my partner Stefanie: Your unflagging support and love encouraged me to finalize this work.

Abstract

Cyber-Physical Systems enable various modern use cases and business models such as connected vehicles, the Smart (power) Grid, or the Industrial Internet of Things. Their key characteristics, complexity, heterogeneity, and longevity make the long-term protection of these systems a demanding but indispensable task. In the physical world, the laws of physics provide a constant scope for risks and their treatment. In cyberspace, on the other hand, there is no such constant to counteract the erosion of security features. As a result, existing security risks can constantly change and new ones can arise. To prevent damage caused by malicious acts, it is necessary to identify high and unknown risks early and counter them appropriately. Considering the numerous dynamic security-relevant factors requires a new level of automation in the management of security risks and requirements, which goes beyond the current state of the art. Only in this way can an appropriate, comprehensive, and consistent level of security be achieved in the long term.

This work addresses the pressing lack of an automation methodology for the security-risk assessment as well as the generation and management of security requirements for Cyber-Physical Systems. The presented framework accordingly comprises three components: (1) a model-based security risk assessment methodology, (2) methods to unify, deduce and manage security requirements, and (3) a set of tools and procedures to detect and respond to security-relevant situations. The need for protection and the appropriate rigor are determined and evaluated by the security risk assessment using graphs and a security-specific modeling. Based on the model and the assessed risks, well-founded security requirements for protecting the overall system and its functionality are systematically derived and formulated in a uniform, machine-readable structure. This machine-readable structure makes it possible to propagate security requirements automatically along the supply chain. Furthermore, they enable the efficient reconciliation of present capabilities with external security requirements from regulations, processes, and business partners. Despite all measures taken, there is always a slight risk of compromise, which requires an appropriate response. This residual risk is addressed by tools and processes that improve the local and large-scale detection, classification, and correlation of incidents. Integrating the findings from such incidents into the model often leads to updated assessments, new requirements, and improves further analyses. Finally, the presented framework is demonstrated by a recent application example from the automotive domain.

Zusammenfassung

Cyber-physische Systeme ermöglichen zahlreiche moderne Anwendungsfälle und Geschäftsmodelle wie vernetzte Fahrzeuge, das intelligente Stromnetz (Smart Grid) oder das industrielle Internet der Dinge. Ihre Schlüsselmerkmale Komplexität, Heterogenität und Langlebigkeit machen den langfristigen Schutz dieser Systeme zu einer anspruchsvollen, aber unverzichtbaren Aufgabe. In der physischen Welt stellen die Gesetze der Physik einen festen Rahmen für Risiken und deren Behandlung dar. Im Cyberspace gibt es dagegen keine vergleichbare Konstante, die der Erosion von Sicherheitsmerkmalen entgegenwirkt. Hierdurch können sich bestehende Sicherheitsrisiken laufend ändern und neue entstehen. Um Schäden durch böswillige Handlungen zu verhindern, ist es notwendig, hohe und unbekannte Risiken frühzeitig zu erkennen und ihnen angemessen zu begegnen. Die Berücksichtigung der zahlreichen dynamischen sicherheitsrelevanten Faktoren erfordert einen neuen Automatisierungsgrad im Management von Sicherheitsrisiken und -anforderungen, der über den aktuellen Stand der Wissenschaft und Technik hinausgeht. Nur so kann langfristig ein angemessenes, umfassendes und konsistentes Sicherheitsniveau erreicht werden.

Diese Arbeit adressiert den dringenden Bedarf an einer Automatisierungsmethodik bei der Analyse von Sicherheitsrisiken sowie der Erzeugung und dem Management von Sicherheitsanforderungen für Cyber-physische Systeme. Das dazu vorgestellte Rahmenwerk umfasst drei Komponenten: (1) eine modelbasierte Methodik zur Ermittlung und Bewertung von Sicherheitsrisiken; (2) Methoden zur Vereinheitlichung, Ableitung und Verwaltung von Sicherheitsanforderungen sowie (3) eine Reihe von Werkzeugen und Verfahren zur Erkennung und Reaktion auf sicherheitsrelevante Situationen. Der Schutzbedarf und die angemessene Stringenz werden durch die Sicherheitsrisikobewertung mit Hilfe von Graphen und einer sicherheitsspezifischen Modellierung ermittelt und bewertet. Basierend auf dem Modell und den bewerteten Risiken werden anschließend fundierte Sicherheitsanforderungen zum Schutz des Gesamtsystems und seiner Funktionalität systematisch abgeleitet und in einer einheitlichen, maschinenlesbaren Struktur formuliert. Diese maschinenlesbare Struktur ermöglicht es, Sicherheitsanforderungen automatisiert entlang der Lieferkette zu propagieren. Ebenso ermöglicht sie den effizienten Abgleich der vorhandenen Fähigkeiten mit externen Sicherheitsanforderungen aus Vorschriften, Prozessen und von Geschäftspartnern. Trotz aller getroffenen Maßnahmen verbleibt immer ein gewisses Restrisiko einer Kompromittierung, worauf angemessen reagiert werden muss. Dieses Restrisiko wird durch Werkzeuge und Prozesse adressiert, die sowohl die lokale und als auch die großräumige Erkennung, Klassifizierung und Korrelation von Vorfällen verbessern. Die Integration der Erkenntnisse aus solchen Vorfällen in das Modell führt häufig zu aktualisierten Bewertungen, neuen Anforderungen und verbessert weitere Analysen. Abschließend wird das vorgestellte Rahmenwerk anhand eines aktuellen Anwendungsfalls aus dem Automobilbereich demonstriert.

Table of Contents

1 Introduction

The advancing digitization drives industries to interconnect previously isolated control systems, enabling countless new use cases and business models such as connected vehicles, the Smart (Power) Grid, and the Internet of Things (IoT).

Such control systems, where computer-aided algorithms executed on an Electronic Control Unit (ECU) interact with the physical world by steering actuators and receiving information from sensors by communication networks, as shown in Figure 1.1, are called Cyber-Physical System (CPS).

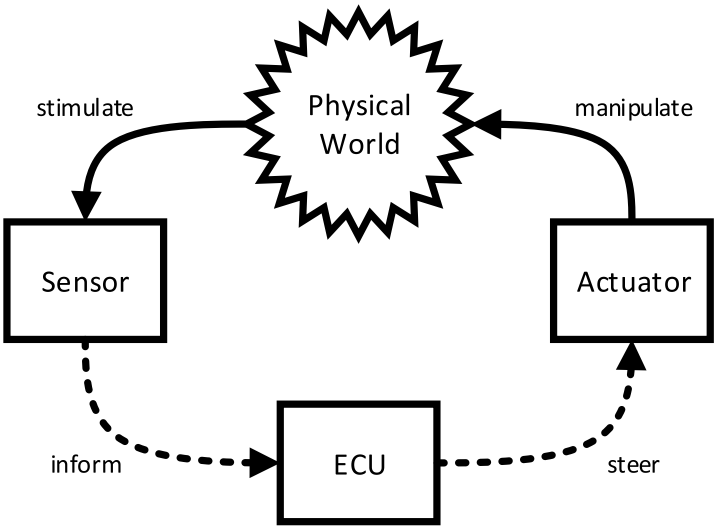


Figure 1.1: The Cyber-Physical Control Flow

Heterogeneous systems, loose topologies, flexible dataflows, and changing entities make the security of CPSs in critical use cases an increasingly challenging matter. The mitigation and prevention of risks caused by malicious activities is the subject of security. Compared to many other domains, security is subject to constant change. While the laws of physics provide a strong constraint in the physical realm, there is no such constraint in cyberspace. In the safety domain, the first occurrence of an exploitable flaw has no direct impact on other installations. In the security area, on the other hand, the risk increases significantly from the first occurrence until the fault is corrected, as attackers often reuse and share their knowledge and tools.

One of the biggest threats to previously secure systems are changes in infrastructure, processes, and dependencies that affect security needs. Attacks on vehicles [1, 2, 3], industrial sites [4, 5], and critical infrastructures like water plants [6] and power grids [7] demonstrate the need for secure CPSs to prevent harm and losses. Most of these incidents were caused by insecure protocols and a lack of access control mechanisms that were not considered during the development of the attacked systems. Nonetheless, the permanent emergence of new vulnerabilities - such as Heartbleed [8], Spectre [9], or in the key exchange of a state-of-the-art Wi-Fi standard [10], enables new attack vectors that can outdate security concepts at any time. Even worse, the likelihood that such a vulnerability will be exploited in repeated attacks increases.

By conducting a Security Risk Assessment (SRA), potential risks are identified and analyzed, highlighting the need for protection. A key challenge in securing CPSs is to determine the appropriate level of protection regarding what needs to be done by what effort and to recognize the need for alterations. Currently, the results of an SRA are used as information for the manual execution of follow-up tasks. Two such important tasks are the synthesis of appropriate security requirements and the validation if such security requirements are fulfilled by a single or a set of CPSs, e.g., in a production line.

The systematic management of the manifold influencing factors requires a degree of automation that exceeds the current state of the art. Currently, many CPS-based domains undergo a paradigm change from dedicated field busses towards more dynamic, service-oriented architectures like the Scalable service-Oriented MiddlewarE over IP (SOME/IP), OLE for Process Control – Unified Architecture (OPC-UA), and the Volkswagen infotainment Web interface protocol (ViWi) [11]. This creates new attack surfaces. Modern cars already have several hundred different security-relevant functions, e.g., for navigation, driver assistance, and motion management. While the number of installed ECUs starts to decrease, the growing number of provided functions increases the complexity and performance of the remaining units. Many of these functions and their hosting ECU are attractive targets for hackers, whether for tuning, denial of service, illegal function activation, or even to harm road users, including but not limited to the driver and passengers.

Agile production lines in the Industrial Internet of Things (IIoT), sharing infrastructures among different companies enable new IIoT use cases, but also introduce unprecedented dynamics to the management of security requirements, especially regarding capabilities and information flows. A particular challenge in the manufacturing industry is matching such requirements with capabilities provided by the operational technology in production lines. Today, most production lines use industrial automation systems such as the SIMATIC S7-300, whose cybersecurity is based solely on segmentation, i.e., the inaccessibility of their network.

Comprehensive and consistent security concepts are necessary to ensure appropriate protection. Their effectiveness must constantly be monitored and adapted to the current situation in order to justify the resulting costs for security controls and measures. While traditional security concepts from the business IT were not successfully adopted to CPSs yet, the industrial oriented ISO/IEC 62443 [12, 13, 14] series and upcoming automotive cybersecurity standards [15, 16] receive increasing attention. All of them demand the implementation of security management processes, including a systematic SRA for the entire product lifecycle.

Knowing the potential risks directly influences the future course of action as it facilitates decisions and helps to avoid unnecessary effort. While it is practically impossible to consider all future possibilities of impairment, it is possible to estimate the potential impacts and make adaptable assumptions about their probabilities by an SRA.

Such an SRA plows the ground for the derivation of intrinsic security requirements that must be met to ensure the proper and secure operation of the CPS as it enables the identification and assessment of valuable assets, their dependencies, and corresponding risks. Up to now, intrinsic security requirements for CPSs are usually formulated manually by security officers, function owners, or component managers. In addition to intrinsic requirements for the protection of the CPS itself, external entities introduce further security requirements regarding functions, techniques, and processes that must be processed and have their fulfillment validated. One challenge addressed by this work is finding an appropriate security level for what needs to be done and at what cost.

After familiarizing with the risks, it must be ensured that their occurrence and changes in the security landscape are adequately recognized. For an appropriate response, it is vital to recognize the signs of the times and to understand what is going on and what consequences are possible. Thus, incident detection, analysis, and correlation are needed to determine whether security requirements have been missed or security goals have been violated. While network monitoring is a common practice to detect incidents in local business networks, it is rarely performed in CPS-environments due to bandwidth limitations, data confidentiality, and sensitive legacy components. To uncover large-scale campaigns against entire supply chains, it is necessary to classify incidents and exchange threat information. Thereby it is crucial to prevent the leakage of valuable business information in order to avoid further losses and market distortion.

The main objective in the field of security is to avoid high risks and problematic surprises. To achieve this objective, it is necessary to systematically identify potential risks, treat them with viable measures, and recognize their materialization.

1.1 Scope

Knowing the security risks, managing the security requirements to mitigate those risks, and assessing and responding to change are important and closely related tasks in maintaining secure systems. In contrast to comparatively standardized enterprise Information and Communications Technology (ICT) systems, specialized CPSs cannot yet be secured by Commercial-off-the-Shelf (COTS) solutions but require customized measures. The overall objective of this thesis is to **improve the risk-based security requirements management for Cyber-Physical Systems** by putting academic methods into practice. The resulting semi-automation greatly supports human analysts. This objective is addressed by the following research questions:

RQ 1 How can potential security risks for Cyber-Physical Systems be systematically identified and assessed in an automated way?
This question is addressed by a model-based security risk assessment in Chapter 4. Thereby, the task is split into creating a suitable model and properly evaluating it.

RQ 2 How to automatically formulate security requirements that support the achievement of the indicated level of protection?
This question addresses two major aspects: How to generally formulate machine-readable security requirements and how to synthesize appropriate requirements, based on the results from the SRA. The answer to this question is subject of Chapter 5.

RQ 3 How to match extrinsic security requirements with the provided security capabilities?
Besides these intrinsic requirements, stakeholders often demand the implementation of external security requirements for new processes or regulations. This is also addressed in Chapter 5.

RQ 4 How to detect, analyze and respond to potentially disruptive changes of the security situation?
Changes in the security landscape must be identified, analyzed regarding their nature, scope and impact, and reflected in an updated SRA. Tools and methods that address this question are subject of Chapter 6.

Not in the scope of this thesis are the practical implementation of the formulated requirements, their testing, and the operation of a Security Operations Center (SOC).

1.2 Impact

This work delivers methods to support the automation of security risks and requirements management, including incident analysis. For these endeavors, the following peer-reviewed contributions have been published.

1. **G. Hansch**, P. Schneider, and G. S. Brost: **Deriving Impact-driven Security Requirements and Monitoring Measures for Industrial IoT**. In *5th ACM Cyber-Physical System Security Workshop*, CPSS'19, Auckland, New Zealand, July 2019, ACM [17].
   Summary: An interactive cybersecurity impact assessment method to determine the individual protection needs of assets as security requirements, and a catalog of countermeasure recommendations.

Context: This work was a contribution to the national research project IUNO Insec [18], in cooperation with Peter Schneider, who contributed a catalog of implementation and configuration recommendations, tailored to the IIoT domain. Furthermore, Gerd Brost contributed a practical use case from the Industrial Data Space (INDS) project.
Own Contribution: As lead author, my contributions to this work are the cybersecurity impact assessment and the requirements derivation methods, which form the basis of Section 4.3 and 5.3.

2. D. Angermeier, K. Beilke, **G. Hansch**, and J. Eichler. **Modeling Security Risk Assessments**. In *17th escar Europe – Embedded Security in Cars*, ESCAR'19, Stuttgart, Germany, October 2019, Ruhr-Universität Bochum [19].
Summary: A graph-based security risk assessment method to systematically assess the risk originating from (new) components and functions.
Own Contribution: My contributions to this work are primarily on the systematic terminology of the graph meta-model, the related work, and the writing of the manuscript, which influenced the overall security model and risk assessment methodology in Chapter 4.

3. **G. Hansch**, P. Schneider, K. Fischer, and K. Böttinger. **A Unified Architecture for Industrial IoT Security Requirements in Open Platform Communications**. In *24th IEEE Conference on Emerging Technologies and Factory Automation*, ETFA'19, Zaragoza, Spain, September 2019, IEEE [20]
Summary: A security requirements data model based on OPC-UA that allows for high and fast automation in the heavily heterogeneous landscape of IIoT. An implementation of the data model is provided using the XML-representation of OPC-UA.
Context: This work was a contribution to the national research project IUNO [18], where the developed framework was deployed and evaluated within a reference project realized by 14 industrial partners and 7 research facilities within Germany.
Own Contribution: My contributions to this work are on the design and implementation of the research, the analysis of the results, and the writing of the manuscript, which form the basis of Section 5.2 and 5.4.

4. S. Plaga, N. Wiedermann, **G. Hansch**, and T. Newe. **Secure your SSH Keys! Motivation and Practical Implementation of a HSM-based Approach Securing Private SSH-Keys**. In *17th European Conference on Cyber Warfare and Security*, ECCWS'18, Oslo, Norway, June 2018. Academic Conferences International Limited [21].
   Summary: A comparison of state-of-the-art Hardware Security Modules (HSM) regarding information security threats by vulnerabilities in x86-based computer systems, which enable the extraction of private keys.
   Own Contribution: My contributions to this work are research for corresponding vulnerabilities and solutions, as well as in the preparation of the manuscript.

5. **G. Hansch**, P. Schneider, and S. Plaga. **Packet-wise Compression and Forwarding of Industrial Network Captures**. In *9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, IDAACS'17, Bucharest, Romania, September 2017. IEEE [22].
   Summary: Methods to capture network records from Machine-to-Machine (M2M) communication and forward these in a compressed and privacy supporting way using dynamic lookup-tables.
   Context: This work was a further contribution to the national research project IUNO [18], where the developed methods were deployed and evaluated.
   Own Contribution: As lead author, my contributions to this work are a chapter on capturing, forwarding, and storing network traffic, and the concept of using lookup-tables in order to reduce redundancy by avoiding repeated transmission of sensitive content, which are the basis for Section 6.2.

6. J. Wolf, F. Wieczorek, F. Schiller, **G. Hansch**, N. Wiedermann, and M. Hutle. **Adaptive Modeling for Security Analysis of Networked Control Systems**. *In 4th International Symposium for ICS & SCADA Cyber Security Research*, ICS-CSR'16, Belfast, UK, August 2016. BCS Learning & Development [23].
   Summary: An ontology language specification for adaptive modeling, including an appropriate refinement and expansion method. Based thereupon, a method to check for known vulnerabilities from various sources during the security analysis.

Context: This paper was a contribution to the national research project SustainGrid.
Own contribution: My contributions to this work are on the adaptive ontology and the related work, which influences Section 4.2.

7. **G. Hansch**, M. Hutle, and W. Fitzgerald. **Smart Grid Threat Analysis using an Attack Tree and Semantic Threat Graph Hybrid**. Poster presented at *Workshop on European Smart Grid Cybersecurity: Emerging Threats and Countermeasures*, Belfast, UK, August 2016 [24].
   Summary: By Attack Trees [25] threats are identified at high levels of abstraction before delving into using Semantic Threat Graphs [26] to identify low-level information about a particular threat under consideration, its impact on corresponding systems, and recommended countermeasures.
   Own contribution: As lead author, my contribution to this work is a multi-stage concept for Smart Grid threat analysis and a description of how to apply this approach.

8. K. Böttinger, **G. Hansch**, and B. Filipovic. **Detecting and Correlating Supranational Threats for Critical Infrastructures**. In *15th European Conference on Cyber Warfare and Security*, ECCWS'16, Munich, Germany, July 2016. Academic Conferences International Limited [27].
   Summary: A threat detection and correlation approach combining machine learning for fine-grained low-level information classification and semantic reasoning for large-scale, simultaneous threat correlation at multiple sites. Unlinkability is preserved by several layers of abstraction while necessary information is shared on a need-to-know basis.
   Context: This paper was a contribution to the ECOSSIAN EU FP7 research project.
   Own Contribution: My contribution to this work is the concept of using different layers of granularity in a hierarchic ontology to communicate threat information, which forms the basis for Section 6.3.

9. P. Wagner, **G. Hansch**, KH. John, C. Konrad, J. Bauer, and J. Franke. **Applicability of Security Standards for Operational Technology by SMEs and Large Enterprises**. In: *25th IEEE Conference on Emerging Technologies and Factory Automation*, ETFA'20, Vienna, Austria, September 2020, IEEE [28].
   Summary: An analysis on the applicability of international standards for the cyber security of operational technology systems regarding company sizes. The feasibility of these standards was analyzed, compared, and assessed by two independent surveys on the topic. As one finding from this investigation is a gap between OT and security experts, we introduced the relevant domain concepts to both sides.
   Context: This paper was a contribution to the national OT-Sec research project.
   Own Contribution: My contributions to this work are on the writing and editing of the manuscript as well as the underlying research, with a focus on establishing a common understanding between operational technology and security.

In addition to these peer-reviewed publications, strong contributions were made to each of the following publications:

- M. Hutle, **G. Hansch**, W. Fitzgerald, T. Hecht, E. Piatkowska, and P. Smith. **D2.2 Threat and Risk Assessment Methodology**. Deliverable, SPARKS Consortium, September 2015 [29].

- H. Sandberg, A. Teixeira, E. Piatkowska, M. Findrik, P. Smith, M. Hutle, and **G. Hansch**. **D2.3 Tools for Smart Grid Cyber Security**. Deliverable, SPARKS Consortium, March 2016 [30].

- R. Chabukswar, A. E. Mady, Y. Hamdaoui, M. Boubekeur, N. Wiedermann, **G. Hansch**, M. Hutle, A. Teixeira, and H. Sandberg. **D2.6 Smart Grid Vulnerability and Risk Assessment**. Deliverable, SPARKS Consortium, March 2016 [31].

1.3 Novelty and Contributions

The automation of security risks and requirements management, taking into account the various threats and dynamic security requirements from different sources, is no individual problem of the CPS domain but affects the entire ICT sector. While corporate IT has reached a certain level of protection in recent years, there is still a long way to go in the CPS domain due to numerous factors such as highly restricted and specialized systems. To this end, this thesis presents a novel framework that combines model-based security risk assessment, requirements derivation, management, and the inclusion of incidents for CPS. The novelty thereby is the direct deduction of security requirements from a semi-automated, model-based security risk assessment with simultaneous support of situation-specific adjustments. In doing so, the following contributions lead to a new level of security risks and requirements management that surpasses the current state-of-the-art.

Contribution 1: A semi-automatable, graph- and model-based security risk assessment method, including a sustainable and machine-readable meta-model in Chapter 4 and Section 6.4.

Contribution 2: Methods for the automated deduction, prioritization, communication and fulfillment-validation of security requirements, supporting a unified security requirements model in Chapter 5.

Contribution 3: A set of tools and methods to support local incident detection and classification, remote correlation and inter-organizational exchange of Indicators of Compromise (IoC) in Chapter 6.

1.4 Structure of the Thesis

This work consists of seven chapters containing the following contents:

Chapter 1: Introduction introduces the current situation, pointing to the overall problem statement that is addressed by the research questions. Also emphasized are the overall scientific impact in the form of scientific publications, the novelty, the contributions, and this description of the structure of the work.

Chapter 2: Foundations is dedicated to the background on CPS-security, security risk assessment, requirements derivation, semantic modeling, and incident management. Furthermore, application domain-specific security aspects and exemplary use cases are presented and discussed.

Chapter 3: Related Work describes the state of the art and presents relevant work on the topic and content of this thesis.

Chapter 4: Model-based Security Risk Assessment introduces a graph-based methodology for assessing security risks, based on the systematic analysis of the impact and probability of a threat to the identified security goals. To enable the creation of the required models, a security-specific modeling language is introduced. This machine-readable presentation forms the basis for subsequent automation.

Chapter 5: Derivation and Management of Security Requirements starts with a formalization of security requirements, supporting two different applications. First, intrinsic security requirements are automatically derived from the assessed security risks using patterns. Second, the propagation of extrinsic security requirements and the validation of their fulfillment are automated.

Chapter 6: Threat Detection, Correlation, and Response deals with the semi-automated handling of threats. First, by a method to support the remote detection of local incidents in resource-constrained CPS, second, by a machine learning-based method for their classification, and third, by their inter-organizational exchange for the correlation of cyber threats on a large scale. A fourth element, which narrows the gap between theory and practice, are recommendations for the reflection of changes in the security landscape into the model from Chapter 4.

Chapter 7: Use Case Evaluation demonstrates the application and usefulness of the methods and processes introduced in the preceding chapters by the evaluation of a recent application example from the automotive domain.

Chapter 8: Conclusion and Future Work contains the conclusion of this thesis, a discussion of the answers to the raised research questions, and an outlook on future research directions in the field of risk-centric security automation for CPS.

2 Foundations

This chapter introduces the terminology, definitions, and basic concepts used in this thesis. Thus, it is dedicated to the background of CPS security, security risk assessment, requirements derivation, semantic modeling, and incident management. Furthermore, application domain-specific security aspects are discussed, accompanied by example use cases.

The start makes an introduction to the Cyber-Physical System security situation and lifecycle in Section 2.1. Continuing with the standardized and normalized process, security risk management and sources for security-relevant information are introduced in Section 2.2. A core element of security risk management is the risk assessment, which is explained in Section 2.3, including threats and attack vectors on the one hand, and on the other, their mitigation by security assumptions, controls, and measures. Models that enable such analysis in an automated, model-based way are the subject to Section 2.4. Security requirements are then introduced in Section 2.5, while domain specific security priorities, in particular of the automotive, critical infrastructure, and manufacturing domains are subject to Section 2.6.

2.1 Cyber-Physical System Security

Cyber-Physical Systems consist of an ECU that executes a computer-based algorithm, which uses communication networks to control actuators that manipulate physical processes, based on sensor readings (cf. Figure 1.1), which leads to the following definition:

Definition 2.1.1 (Cyber-Physical System)**.** A control system in which a computer-based algorithm utilizes one or multiple communication networks to connect sensors, actuators, and ECUs in order to interact with the physical world.

The lifecycle of systems can be divided into a development, operation, and decommission phase, each of which can be refined into various levels of detail. For security-relevant systems, this lifecycle process should be augmented by an initial training of the involved humans, and a response phase, as outlined in Figure 2.1.

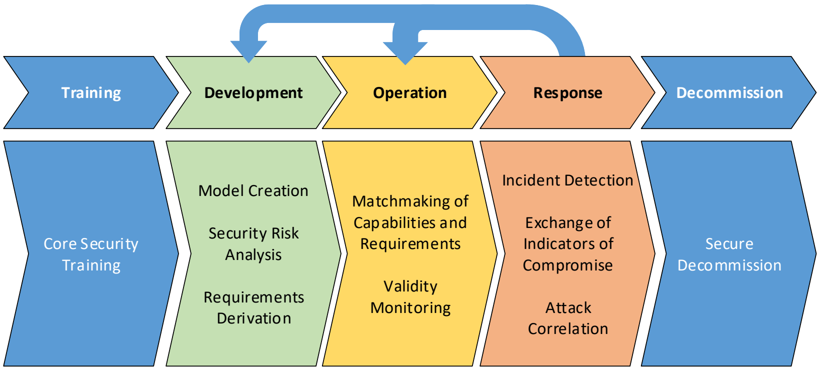


Figure 2.1: High-level Phases of the Secure System Lifecycle Process

A well-known refinement for the development phase of this process is the *Security Development Lifecycle Process*, shown in Figure 2.2 [32, 33]. Deviating from the order of this process, the threat modeling in Chapter 4 is conducted in advance of the security risk assessment, for which it provides an essential input factor.

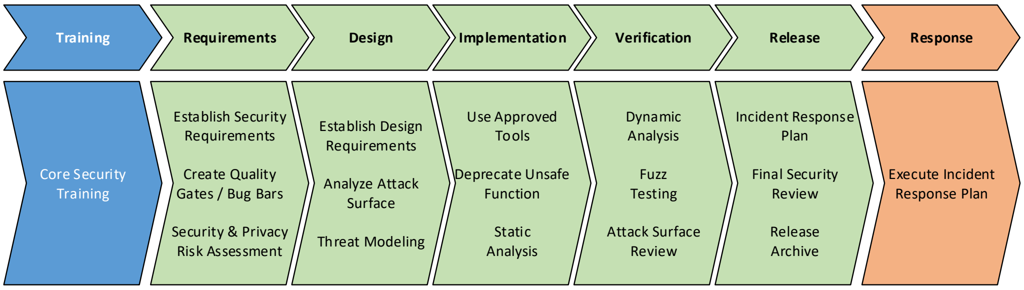


Figure 2.2: Phases of the (Microsoft) Security Development Lifecycle Process [33]

Due to its durable environment, the usual lifecycle of a CPS is significantly longer than the three-year average of ICT systems in enterprise and consumer environments. For the automotive and industrial domains, three to six years of development time and ten to fifteen years of operation are no unusual assumptions [34, 35]. Concerning operation times of CPSs in critical infrastructures, no reliable data is publicly available. An indication may be the operation time of up to 60 years for power transformers [36].

Definition 2.1.2 (Asset)**.** All items of tangible or intangible value for a stakeholder are assets.

Examples of tangible assets are components, physical connections, and complete control systems, while reputation and intellectual property are intangible.

Definition 2.1.3 (Security Objective)**.** Security objectives are the confidentiality, integrity, availability, authenticity of assets, and the non-repudiation of actions.

Definition 2.1.4 (Security Goal)**.** A security goal is the preservation of a security objective.

A security goal can depend on another security goal, such as the availability of data depends on the availability of the component where it is stored.

Definition 2.1.5 (Attack Goal)**.** An attack goal is the intended violation of a security objective by a malicious attacker.

During operation, but also development, there is a risk of attacks leading to a compromise of information security assets. Adapting security to the dynamics of rapid technology changes requires reiterating time- and resource-intensive threat and risk assessments. Nevertheless, changing business needs and the security landscape require additional information and short-term adjustments. The long lifespan further demands a more comprehensive view of the operating phase as opposed to comparatively short-lived consumer devices and software. Further problems in maintaining CPSs security are cost optimizations, avoiding spare resources, and their distribution in terms of locations, making upgrading security functions such as encryption algorithms or replacing certificates a challenge. Besides this, high demands on availability and integrity allow only minimal maintenance interruptions, which must not fail and result in a non-functioning unit.

To this aim, the foundation for CPS-security should be laid as part of the system engineering process, rather than by retrofitting them at a later date. Security engineering as enabler for the construction of dependable distributed systems is subject of an increasingly amount of standardization for both CPS specific technologies [37] and application domains [12, 16]. These standards provide the formal ground for a continuous security management of connected vehicles, the Smart (power) Grid, and the IIoT. Among the first tasks of security engineering are the identification of assets, assessment of their security objectives, and derivation of their relevant security goals.

2.2 Security Risk Management

Security Risk Management is the process of identifying, analyzing, and managing security risks. An introduction and overview on this topic are provided by the ISO/IEC 27000:2018 “Information technology – Security techniques – Information security management systems – Overview and vocabulary” standard [38]. It introduces a basic process model, a common taxonomy, and the further ISO 27000 series. The process to manage information security risks intentionally caused by malicious attackers is subject of the ISO/IEC 27005:2018 “Information technology – Security techniques – Information security risk management” standard [39]. The risk management process described there is illustrated in Figure 2.3. Considering risk management on an abstract level, principles and guidelines that can be used within these process models are subject of the ISO 31000:2018 “Risk management – Guidelines” standard [40], which defines risk management as “coordinated activities to direct and control an organization with regard to risk”. Focal points of the defined process are the assets that are critical to the operation of an organization.

The ISO/IEC 27005:2018 specifies the following output for the individual steps of a risk management process:

Context Establishment The specification of basic criteria, the scope and boundaries, and the organization for the information security risk management process.

Information Security Risk Assessment A list of assessed risks prioritized according to risk evaluation criteria.

Information Security Risk Treatment Risk treatment plan and residual risks subject to the acceptance decision of the organization’s managers.

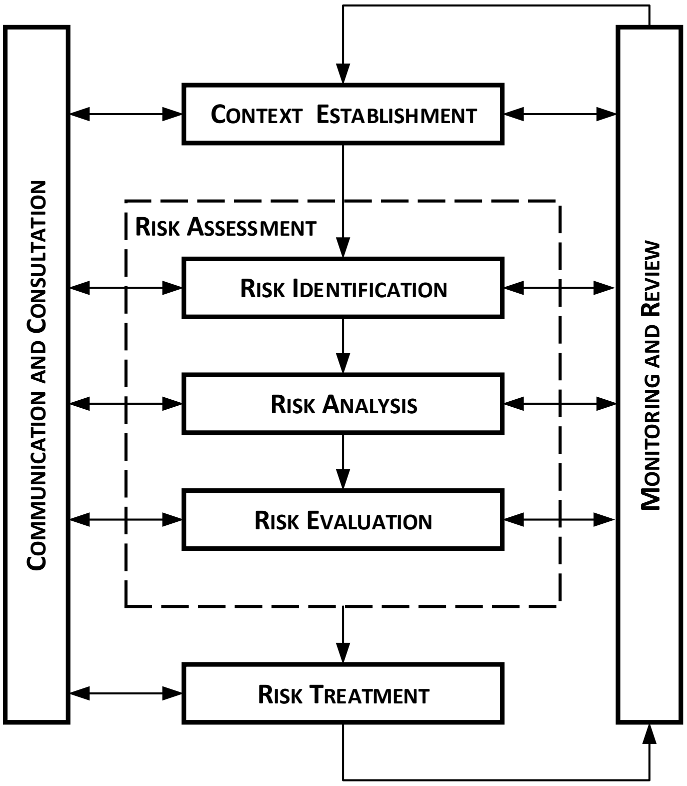


Figure 2.3: Risk Management Process from ISO/IEC 27005:2018 [39]

Information Security Risk Communication and Consultation Continual understanding of the organization's information security risk management process and results.

Information Security Risk Monitoring and Review Continual alignment of the management of risks with the organization's business objectives, and with risk acceptance criteria.

A fundamental part of most security management strategies is the information security risks identification during development and operation [39, 41, 42]. The practical identification and assessment of valuable assets is an interdisciplinary challenge that requires the collaboration of application and security experts.

Sources of Security-related Information

Public databases and targeted information exchange of security-relevant content make valuable contributions to cross-organizational CPS security.

Common Attack Pattern Enumeration and Classification (CAPEC) The CAPEC [43] is a comprehensive dictionary and classification taxonomy of known attacks, maintained by MITRE. It can be used by analysts, developers, testers, and educators to advance community understanding and enhance defenses.

Vulnerability Databases Typically maintained by a SOC, vulnerability databases provide information about known vulnerabilities in products. While many organizations, domains, and nations maintain private databases, prominent public examples are the NIST National Vulnerability Database (NVD) [44], and the SecurityFocus Vulnerability Database [45] that is based on the Bugtraq mailing list. When searching for vulnerabilities of programs and components, it is essential to have references to their corresponding entries in related databases.

Common Weakness Enumeration (CWE) The CWE [46] is an enumeration of software weaknesses that might result in the vulnerability of a product. Just as for vulnerabilities, the creation and linking of such databases [47] support various forms of automated security analysis and penetration tests [48].

Common Platform Enumeration (CPE) The CPE [49] provides a structured naming scheme, based upon the generic syntax for Uniform Resource Identifiers (URI), for information technology systems, software, and packages. Maintained by NIST, it provides a formal name and description format as well as methods for checking names against a system, and to bind text and tests to a name. Although it is not primarily focused on security, it provides valuable assistance in mapping vulnerabilities and weaknesses to the relevant products. It can be used as a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among computing systems.

Threat Intelligence Another potent source of security-relevant information is threat intelligence. Platforms such as conferences, forums, repositories, and unpublished documents [50] provide further valuable information about vulnerabilities and threats.

Targeted Information Exchange Beside the public databases, further security information exchange processes such as limited disclosure, including early warnings, are common for critical infrastructures.

Security Best-Practice List Best-Practice lists, as the OWASP *Application Security Verification Standard Project* [51] and the *Proactive Controls* [52] provide up-to-date recommendations to avoid common mistakes and maintain a high level of security. While exemplary recommendations for security controls are given in Chapter 5, these projects should be preferred for more detailed and up-to-date recommendations.

2.3 Security Risk Assessment

As part of the security risk management, a Security Risk Assessment (SRA) provides qualified information about the security risks of a System under Evaluation (SuE). Knowing the potential risks strongly supports various further security-related activities like the derivation of requirements (cf. Section 5.3) and systematic testing, e.g., by fuzzing [53, 54].

Risk is usually calculated as the product of likelihood and impact, as shown in Equation 2.1. Likelihood thereby refers to the implementation of a threat triggering the impact.

Definition 2.3.1 (Risk)**.** The likelihood of the occurrence of a negative event and the amount of potential damage that can be caused thereby.

$$\text{Risk} = \text{Likelihood} \cdot \text{Impact} \tag{2.1}$$

An SRA should be performed early in the design process but is also strongly advisable for existent systems. As the handling of identified threats often causes changes to the design or operation, it is beneficial to find them as early as possible. Typical ways therefore are catalog- and model-based assessments.

Catalog-based analysis methods typically provide checklists, constraints, and scoring spreadsheets to evaluate a system deterministically. A formerly accepted checklist-based method for evaluating ICT systems was to quantify the risk by a comparison of recommended and implemented controls [55], i.e., if a virus scanner, a mechanical disk lock, or patches are installed. The increasing number of potential attack vectors, combined with fast software update cycles, made such enumerative lists no longer maintainable. Instead, the requirements are formulated more abstractly, such as “ensure access control” or “effectively deactivate interfaces”.

A model-based analysis is an efficient and systematic way to identify and assess risks and threats. As models are only an abstraction of real systems, they cannot contain all details. Nonetheless, it is essential to hold all information that is required for their designated purpose. A problem with graph-based modeling is the tension triangle of expressiveness, correctness, and simplicity in terms of usability. Simplification can be achieved by abstractions and modularization, respectively encapsulation. While abstractions cause a loss of information, modularization is susceptible to the loss of cross-references.

In contrast to simple systems that can be assessed as one, complex systems require proper scoping and abstraction. Often there are significant differences between the domains. While entire business use cases in the Smart Grid [29] and IIoT [17] domains could be successfully assessed all at once, analyses in the automotive sector greatly benefit from scoping to individual functions or components due to the required level of detail.

2.3.1 Threats and Attack Vectors

An important part of continuous security risk management is the connection between threat, risk, and vulnerability analysis.

Definition 2.3.2 (Threat)**.** Any circumstance or event with the potential to adversely impact an asset through unauthorized access, destruction, disclosure, modification, or denial of service.

This threat definition is a slightly generalized version of the ENISA definition [56], with a lift of the limitation of modification from data, because functions and physical components might also be altered.

Definition 2.3.3 (Threat Actor)**.** The source of security threats, also called an attacker that may have a malicious intention to violate a security goal. This entity can be an individual but also an organization.

Definition 2.3.4 (Threat Factor)**.** A characteristic, which a threat actor must provide to perform a specific task.

Common threat factors are the required time, expertise, knowledge of the SuE, window of opportunity, and equipment required for exploitation [41].

A threat analysis aims to identify (and assess) the causes of malicious incidents that may harm a system or organization. Identification and assessment of potential security risks are subject to risk analysis. A vulnerability analysis is used to identify and evaluate weaknesses that can be used by a threat actor to trigger a risk. It can be conducted during the risk analysis, but also be a downstream task. Since vulnerabilities become known after the release, they are an essential factor that can change the required potential to trigger a risk, leading to an updated risk assessment. Conventional methods to find vulnerabilities are audits, dedicated scanners for known vulnerabilities, and penetration testing, including fuzzing and reverse engineering.

The threat actor exploits a weakness to violate a security goal. The likelihood of such an attack can be calculated by a comparison of costs and benefits, where costs represent the required effort. According to [57], the choice of an attacker to commit such an action can be calculated by Equation 2.2.

$$M_{\mathrm{b}} + P_{\mathrm{b}} > O_{\mathrm{cm}} + O_{\mathrm{cp}} \cdot P_{\mathrm{a}} \cdot P_{\mathrm{c}} \tag{2.2}$$

The left side of this equation covers the monetary benefits M_{b} and psychic benefits P_{b} of the attacker, while the right side covers the costs for the monetary opportunity O_{cm}, the psychic opportunity O_{cp}, the probabilities of apprehension P_{a}, and of conviction P_{c}.

Security-domain specific problems are thereby the disruptive changes in terms of costs, especially between first and repeated compromise, and the unpredictability of benefits from the attacker's point of view. While finding and exploiting a vulnerability can take a significant amount of time, require expensive equipment, and specialized knowledge, following a tutorial to execute a published exploit is an easy and cost-effective way. Even more challenging is the quantification of benefits. While benefits from fraud, espionage, and denial of a competitor might be quantified to some degree, immaterial motivations such as fame and “fun by causing mayhem”, which were claimed by the hacking group “LulSec” for highly sophisticated attacks [58], cannot be rated reasonably. In practice, this situation is often solved by restricting the considered attack models to those with likely benefits, such as the owner of a vehicle for manipulations requiring physical access to the inside of the vehicle.

Definition 2.3.5 (Attack Vector)**.** A path or combination of actions taken by an attacker to violate a security goal.

The systematic enumeration of potential attack vectors supports the understanding of interdependencies between different attacks and allows the identification of relevant weaknesses.

Attacker Models

Attacker models can be differentiated into attacker- and capability-centric models.

Attacker-centric Models consider a set of different attackers, each with a fixed set of properties and (immutable) capabilities [59, 60], e.g., that the attacker is a legitimate user of the network and can initiate a communication. In times of video guides, publicly available exploits, and inexpensive devices such limitations are rarely maintainable.

Capability-centric Models estimate the capabilities an attacker must have to execute a threat successfully. Typically considered capabilities are the required time, expertise, equipment, and knowledge of the attacker [41]. These capabilities can be standardized in an organization-specific assessment model and used as input constants for the automated generation of security requirements [19].

Weakness, Vulnerability, and Indicator of Compromise

Security relevant mistakes in the design of a system are considered weaknesses.

Definition 2.3.6 (Weakness)**.** A weakness is a type of mistake in a design that, under appropriate conditions, could contribute to the introduction of vulnerabilities within its implementation. This term applies to mistakes regardless of the phase they are introduced.

Vulnerabilities represent a materialization of design weaknesses and can significantly reduce the effort required by an attacker to violate a security goal. Discovered vulnerabilities typically require a reassessment of the current risk situation.

Definition 2.3.7 (Vulnerability)**.** A vulnerability is the occurrence of one or more weaknesses in an implementation that enables a threat to compromise its security objectives.

Definition 2.3.8 (Indicator of Compromise)**.** An Indicator of Compromise (IoC) is forensic evidence of intrusion by exploiting a weakness or vulnerability that can be identified on a host or network, e.g., by a log-file entry.

The granularity of possible IoCs ranges from low-level field signals to an incident report filed by a security analyst.

2.3.2 Threat Mitigation - Assumption, Measure, and Control

To mitigate threats, various approaches are possible. Basically, they are divided into assumptions, measures, and controls.

Definition 2.3.9 (Security Assumption)**.** A security assumption is an expectation that a specific fact, supporting a security goal is and stays true.

A common security assumption is that stare-of-the-art encryption algorithms provide a proper level of security. This assumption is no longer valid and considered broken if decryption without the proper key is made possible by exploiting a weakness.

Definition 2.3.10 (Security Measure)**.** A security measure is an (organizational) precaution against the violation of a security goal.

Definition 2.3.11 (Security Control)**.** A security control is a (technical) practice, procedure or mechanism that aims to avoid, detect, counteract, or minimize security risks.

A key philosophy of the security product lifecycle is the defense-in-depth strategy [61], as it defends the system against any particular attack using several independent methods. It implies different layers of protection and detection, based on the idea that the effort and risk of detection increases as each layer must and might be defeated during an attack. Thereby, flaws on a single level can be mitigated by capabilities on deeper levels.

2.4 System Modeling for Automated Security Analysis

System models are widely used for the development and analysis of non-trivial systems. Their quality directly frames the performance of further processes. Achieving high accuracy and completeness in practice is often the hardest part of the analysis due to incomplete or inaccurate information about the SuE. Thereby, not all parts of the system need to be modeled on the same level of detail.

A good starting point for technical models that can be adapted for security management are component and sequence diagrams. The combination of such diagrams often enables the creation of an overall system model that can be augmented with security-specific information, such as security objectives. Most organizations today use a Domain Specific Modeling Language (DSML) to model systems in their daily business. These DSML are in many cases based on the Unified Modeling Language (UML) (cf. Section 3.1.1), or ontologies, expressed in Web Ontology Language (OWL). Domains that make heavy use of ontologies are, e.g., medicine [62], linked web data [63], and knowledge organization systems. The common purpose and object-orientation of both UML and OWL for knowledge representation lead to numerous syntactical similarities [64]. Managing the complexity of the resulting model, to avoid inconsistencies and flaws, requires a certain degree of automation and can be supported by usage of a machine-readable knowledge representation language. A valid system model in such a machine-readable language opens a multitude of different application possibilities. The resulting knowledge-graph makes implicit relations explicit by definitions of the classes, properties, and their relationships.

To allow experts in the application domain to describe their systems more intuitively, to simplify the integration of existing models, and to infer facts, OWL is used as the modeling language in this thesis. In the language of OWL, statements about resources are typically formatted using the Resource Description Framework (RDF), while the Resource Description Framework Schema (RDFS) is used as a (hierarchical) class schema. The Web Ontology Language (OWL) in its decidable description logic variants (e.g., OWL-DL) supports the creation of formal correct descriptions, their processing, and understanding. Although a separate program can conclude heterogeneously grained information, it is frequently more efficient and less prone to errors to use a semantic reasoner [65, 66] that finds and highlights inconsistencies. Thus, conclusions can be drawn using ontologies with a determined taxonomy and semantic relations enabling entailment relations and to request facts by a semantic query language like SPARQL. To formalize models of SuE, the following formalization based on [23, 67] is used: An ontology in the formal description language OWL 2 DL is expressed in terms of 'concepts', 'roles', and 'individuals'. The membership of an individual $\texttt{I}$ to a concept C is denoted by $C(\texttt{I})$. Roles are partial functions on individuals. A role $r(x, y)$ can be interpreted as a relation between individuals x and y, meaning x has a property r to y. The domain and range of roles are defined by writing them as partial functions ($r : \langle domain \rangle \nrightarrow \langle range \rangle$). If C_1 is a sub-concept of C_2 (i.e. $\forall x : C_1(x) \rightarrow C_2(x)$), this is denoted by $C_1 \sqsubseteq C_2$, and analogue for sub-roles. This hierarchy of concepts is used by a semantic reasoner to infer and extend the model. The operators $\equiv$, $\sqcup$, and $\sqcap$ are concept equality, concept union, and concept intersection, respectively. The concept containing all individuals is denoted by $\top$ while the empty concept is denoted by $\bot$. For a role R and a class C, the class-expression $\exists R.C$ denotes the set of all individuals connected via R to another individual, which is an instance of C. Furthermore, $\forall R.C$ describes the class of all individuals for which all via R related individuals must be instances of C. Number restrictions (like $\exists_{\geq n}$) are used to describe the number of individuals, related to a role. Also, expressions are provided in predicate logic. Variables in predicate logic are written as $?x$. Expressions about classes allow the machine-based reasoner to deduce implicit knowledge that is not explicitly stated. Assume for instance that a concept $C_1 \sqsubseteq C_2$. Then any individual $\texttt{I}$ that is in C_1 is by definition also in C_2, and therefore any statement made on the more general concept C_2 holds also for $\texttt{I}$. In particular, the operations $\equiv$, $\sqcup$, and $\sqcap$ allow construction of new concepts.

2.5 Security Requirements

Security Engineering is the dedicated engineering discipline for the development of secure systems. It is up to this discipline to formulate suitable intrinsic requirements in order to design and operate secure systems. Good requirements are often characterized by the acronym SMART [68], meaning they are **S**pecific, **M**easureable, **A**chievable, **R**easonable and **T**ime bond.

The formal ground for process requirements for the secure development of products used in industrial automation and control systems is the IEC 62443 “Industrial communication networks – Network and system security” standard series that promotes a holistic approach [12, 13, 14, 61].

Since security is a hygiene factor whose absence has negative consequences, but whose presence rarely provides a marketable benefit, costs must be in a reasonable relation to the sum of potential losses. Protection is only adequate if the possible direct or indirect damage outweighs its costs.

In the course of this thesis, security requirements are subject to Chapter 5, where they are distinguished into intrinsic and extrinsic security requirements. Intrinsic security requirements are derived by security engineering, typically based on a security risk assessment, and target the protection of the SuE. Extrinsic security requirements, in contrast, originate from external entities and target to protect processes or implement regulations. While overlaps are advantageous, there may also be contradictions between intrinsic and extrinsic requirements, which must be resolved accordingly.

Definition 2.5.1 (Intrinsic Security Requirement)**.** Intrinsic Security Requirements must be met to maintain the security goals of the implementation system itself.

Definition 2.5.2 (Extrinsic Security Requirement)**.** Extrinsic Security Requirements originate from external influences such as processes and regulations that are introduced in the interest of an external entity. Non-compliance endangers the external interest but not necessarily the intrinsic security goals of the implementation system itself.

2.6 Domain-Specific Security Priorities

Most CPS-using domains share a common set of security goals. They typically aim to prevent damage to life and limb, financial losses, and to preserve property. Severe deviations often exist regarding domain-specific languages, effective regulations, the weighting of security objectives, and operating environments. An essential factor is usually the difference in the respective roles and rights. In particular, it makes a severe difference whether the roles of owner and user of a CPS coincide as with a customer-owned car, or not, as with an operator-owned but customer-used infrastructure. In addition, cultural, regulatory, and geopolitical differences often need to be taken into account, particularly with regard to privacy and communications.

The different weighting of security objectives, concluded from personal experiences during numerous industrial and research security-risk related projects in the course of this work, is sketched in Figure 2.4.

For the automotive sector, these weightings are the conclusion of more than 100 development projects to assess security risks and develop protection concepts for functions and components, carried out for various German automotive OEMs and suppliers.

The basis for the weightings of Critical Infrastructures (CI) and the IIoT originate, primarily from the collaboration in several research projects. For the critical infrastructures, these are the EU FP7 projects SPARKS - Smart Grid Protection Against Cyber Attacks (Grant № 607577) and ECOSSIAN - European COntrol System Security Incident Analysis Network (Grant № 607577), as well as the national IuK-Bayern research project SustainGrid (Grant № IuK423). For the IIoT, they base (in addition to the other sources) on the participation in the IUNO - National reference project for IT-Security in Industry 4.0 (Grant № 16KIS0324), its successor IUNO Insec (Grant № 16KIS0933K) and the national IuK-Bayern research project IT-Sec (Grant № IuK585).

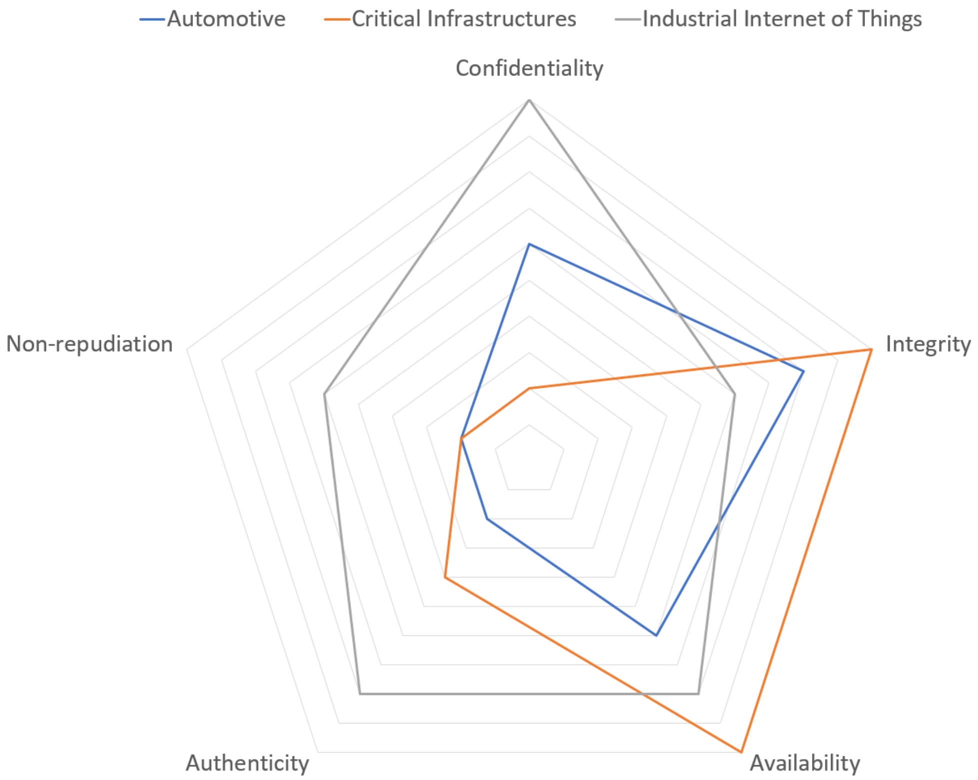


Figure 2.4: Domain Specific Prioritization of Security Objectives

2.6.1 Automotive

Increasing connectivity exposes modern vehicles to cyber-attacks [1]. So far, the implementation of automotive cybersecurity has been based on national and company-specific regulations with strong ties to safety concerns and know-how protection. Emerging harmonized international regulations, especially the UNECE WP.29 GRVA [15] and the ISO/SAE 21434 [16], cause emancipation of cybersecurity as an own discipline besides safety. Among the joint demands of these regulations is the implementation of a systematic security risk assessment and a security management process for the development, production, and post-production phases. A national approach of German automotive manufacturers to improve their supply chain security is the *Trusted Information Security Assessment Exchange* (TISAX) certification [69], by which suppliers must provide unified security assessment audit results, assessed and certified by a trustworthy auditor. Following this approach, the exchange of achieved audit levels shows that appropriate action is taken without revealing abusive attack information in case of a leak. Nonetheless, a valid TISAX certification can indicate a basic level of security.

Typical considerations in the automotive domain are the safety of road users, financial and legal aspects for both the manufacturer and the driver, operational safety concerning reliable, trouble-free operation, confidentiality, privacy, and the image of the brand. Usually considered attackers in the sense of vehicle security are tuners, saboteurs, thieves, and competitors. The use case presented below from the automotive sector will be referred to in the following chapters.

2.6.2 Critical Infrastructures

In critical infrastructures, such as water and power plants, CPSs control processes by pumps, tap-changers, inverters, and robots, making availability and integrity primary security objectives. A major challenge when performing a risk assessment for this domain is the identification of potential physical consequences of a cyber-attack. Such attacks could result in safety-related incidents, but also cascading effects to dependent infrastructures. In addition to the risks posed by enabled cyber-attacks, the connection of legacy systems, some of which have existed for decades, to modern ICT creates additional risks when new security controls are introduced into existing legacy systems. Usual attackers in this domain are thieves, vandals, saboteurs, and competitors.

2.6.3 Industrial Internet of Things

Industrial production line devices are often expensive, long-term investments requiring decades of support and maintenance. The introduction of CPSs into industrial installations means a fundamental shift from isolated production to demand-specific reconfigurations of data flows and production lines, often called the fourth industrial revolution.

While many findings from the discussion about cybersecurity and privacy risk for consumer Internet of Things (IoT) devices [70] are transferable to the industry, the increased life cycles, infrastructures, and values of IIoT require additional planning and efforts.

IIoT-operators demand the ability to propagate security requirements along the supply chain, to validate the compliance of a production line with specified security requirements and to verify in retrospect whether products were manufactured in a compliant process.

Regarding security risk management for Operational Technology (OT), three main aspects can be distinguished.

1. **Static installations**, where only the threat situation with regard to exposure needs to be regularly re-evaluated, but the impact remains constant.
2. **Dynamic installations**, where changes to the OT-infrastructure may alter exposure and impact. Changes should be considered here in advance and must only be implemented if the resulting risk situation is acceptable.
3. **Product-centricity**, in which impacts depend on the goods produced. In this case, threats are constant, but the potential impact vary.

Conventional networked devices serve a specific purpose and have a limited number of communication partners and data flows. Contrary to this, the number of communication links and flows in a dynamic IIoT-network can grow rapidly and significantly with the addition of new sensors, actuators, and ECUs. While recent industrial CPS often already support updating their software and hardware, they still lack appropriate protection for service-oriented architectures (SoA), software-defined networking (SDN), and support for the spontaneous addition or removal of components. The resulting rerouting can suddenly change the current threat situation due to shifting dependencies or exposures. This increases the challenge of ensuring compliance with the security requirements of an infrastructure used jointly by more and more communication partners. It demands an automatism to quickly assess the current situation and provide appropriate security strategies for each new configuration.

The *Industrial Internet of Things Reference Architecture* [71] and the *Reference Architecture Model Industrie 4.0* [72] are two reference architectures, each presenting numerous concepts and methods from different perspectives. Their common goal is to support their users in creating implementation concepts [73] by reaching a common understanding. While [71] concentrates on vertical cross-domain and interoperability, [72] focuses on the horizontal value chain of manufacturing.

Another challenge in assessing IIoT cybersecurity risks are globally distributed, cross-company supply chains. Besides the question on how to handle impairments that affect up- and downstream partners, there is a problematic lack of information as suppliers seldom share ICT security-related information with their customers, which states a significant uncertainty for operational, tactical, and strategic planning. Although this is understandable to protect against attackers and competitors, it complicates the exchange of IoCs and incident information. The ISO 28000 [74] is the normative framework for security management systems for the supply chain. It identifies the establishment and maintenance of procedures for adequate security risk analysis as part of a dedicated security management system. The factors mentioned there are considered to be part of information security within the ISO 27000 series. Taking into account the relevant international standards [38, 40, 74], and domain-specific practices [75], a practical system to assess the probability of different types of threats is required. Inputs for this assessment are (among others) an analysis of publicly available information, supplier-specific attack surfaces, and attack attractiveness.

Usual attackers in this domain are competitors, thieves, and saboteurs.

3 Related Work

This chapter introduces the state of the art and presents relevant work on the topic and content of this thesis.

3.1 Security Risk Assessment

The objective of each security management strategy [38] is dealing with security risks [39, 42]. Conducting an SRA enables the systematic identification of such security risks, from which security requirements can be derived in a subsequent step, as described in Chapter 5. The SRA is typically done either by a model-based risk analysis or for simple systems by spreadsheets. In their survey, the authors of [76] examine and evaluate ten current methods of security engineering regarding their validity, compliance, costs, and usefulness. Similarly, the author of [77] assesses the different metrics of such techniques. A dedicated comparison of SRA methods is provided by [78].

As the name correctly suggests, model-based security risk assessment consists of a modeling and a subsequent assessment phase. Tools to perform attack graph-based security risk assessments are described by [79, 80, 81, 82]. For methods analyzing systems during their operation phase, the collection of information for the model creation can be supported by a network security scanner, i.e., Nessus, and combined with available topology information in a pre-defined class model, in particular, information about how devices are interconnected. A survey of attack and defense modeling approaches that are based on directed acyclic graphs is given by [83].

3.1.1 System Modeling

Model-engineering and ontology-engineering are currently two competing paradigms for the creation of machine-readable system models. While both approaches are comparable regarding the modeling and analysis capabilities relevant in this thesis, the practical application showed that while model-engineering using UML is more common for a computer engineer, experts from non-IT domains find it easier to model systems in a semantic description language like OWL, based on a domain-specific ontology. A detailed comparison of UML and OWL is provided by [64]. In this thesis, most practical implementations and listings are formulated in OWL but visualized as UML diagrams in favor of better readability.

Unified Modeling Language (UML)

The most prevalent and widespread modeling-centric language in the field of computer science is the UML [84], which is not designed for automated, machine-based security analyses but supports language extensions to such aims. Common extensions are SecureUML [85] for the definition of access control rules, and UMLSec [86] for their analysis. SysML focuses on the support of the specification, analysis, design, verification, and validation of systems and systems-of-systems. While these model languages are suitable for the design and development phases, they lack methods to successively refine individual components and to freely describe individual object relationships. Detailed knowledge about the critical components, their parts, and relations is, however, crucial for the analysis.

Ontologies for Security Analyses

A common approach to model systems for subsequent machine-based analysis, avoiding inconsistencies, is to use description logic in the form of an appropriate ontology [79, 87, 88]. Security-centric ontologies are presented by [89, 90]. A good overview of further risk assessment ontologies is provided by [91], whose authors compare 13 ontologies to synthesize an own meta-ontology, combining several high-level security concepts, axioms, and attributes. Targeting requirements engineers as their audience, the authors specify their models in the Web Ontology Language (OWL), reaching a certain level of automation with SQWRL queries [92] and semantic reasoners [65, 66]. Work on machine-based reasoning introduced some degree of automation that can effectively be used as part of an interactive systems engineering process [93], to generate network rules [26], and to locate known vulnerabilities [23, 79] in SuE.

An approach for the automated search for known vulnerabilities in incomplete or inconsistent described systems is described by [23]. The authors, therefore, introduce and demonstrate an ontology-based method to adaptively model and analyze CPS from a security perspective. By using machine-based reasoning on a semantic model of a SuE, further security-relevant information is inferred. They further provide a formalism to handle incomplete information by applying iterative extension and refinement of the model where necessary. Based on this model, non-obvious attack vectors are identified and linked with vulnerability information. This approach allows focusing the modeling on those parts of the system that are relevant for the security analysis, and is partially taken up and further developed in Section 4.2.3.

3.1.2 Threat Modeling

In the context of this thesis, threat modeling is considered the process of identifying potential threats and evaluating their feasibility. The following enumeration describes relevant threat modeling methods in the context of this work.

Attack Trees and Graphs

The canonical example of a structured security risk analysis by a directed graph is the Attack Tree (AT) [25], as shown in Figure 3.1. An AT provides a semi-formal and cost-effective way of structuring the various threats that an asset may encounter. It enables the identification and evaluation of attack vectors by combining several vulnerabilities in one analysis and identifying the most likely attack vector. In practice, these trees are used for threat elicitation and analysis – representing threats at a high-level of abstraction – and have tended not to be used to capture low-level or concrete security configuration detail. However, such an approach suffers from several shortcomings, as they tend to state explosions when rather simple goals are split up in a large subset of intermediate steps, and the risk of missing potential attack vectors or entry points an attacker could use increases. An alternative avoiding such shortcomings, often at the cost of intuitive readability, are graph-based approaches [19, 94], as shown in Figure 3.2. A comprehensive survey of security-related trees and graphs is provided by [83]. One approach to overcome the mutual problems of graphs and trees is the refinement of an initially high-level tree using graphs for reoccurring and complex blocks [24].

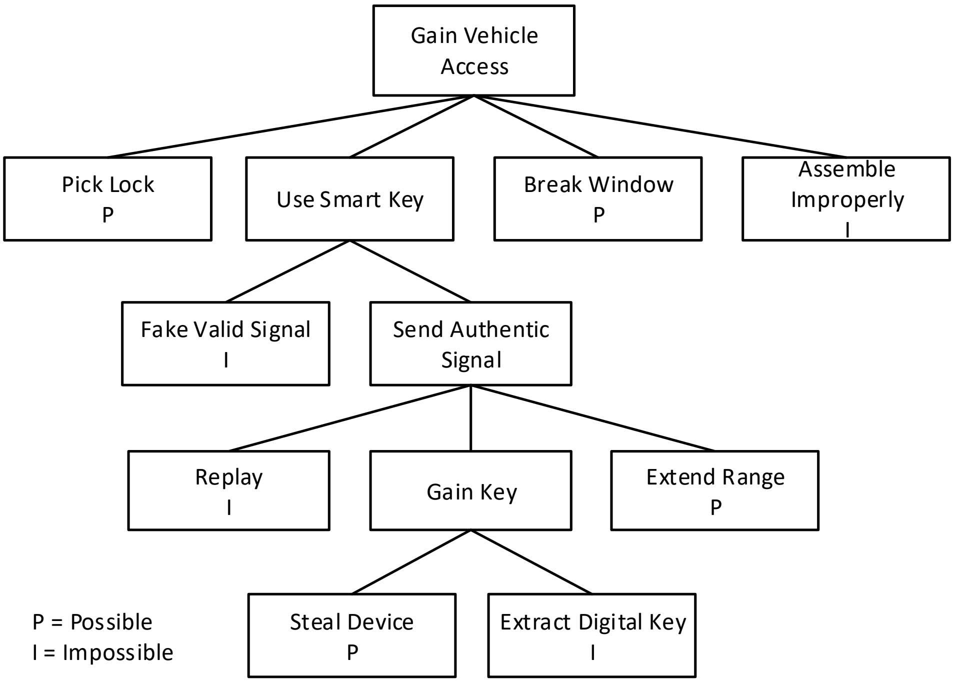


Figure 3.1: Exemplary Attack Tree on Gaining Vehicle Access

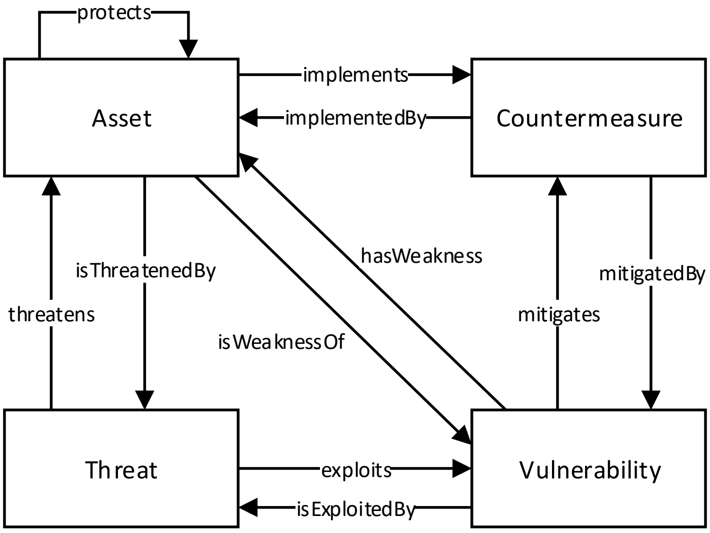


Figure 3.2: The Semantic Threat Graph [94, 29]

Misuse Cases

A common software engineering method for threat modeling, especially when it comes to securing user interfaces, are misuse cases. Developers and analysts, therefore, consider what an attacker's goals might be and by which combination of actions he can achieve them, contrary to the intended system functionality. According to [41], potential reasons for misuse are incomplete or unreasonable guidance, unintended misconfiguration, and forced exception behavior.

As a threat modeling method, misuse cases have an inherent risk of overlooking potential cases due to lack of information, experience, or imagination. Furthermore, attack vectors that are unknown or found impossible at the time of the risk analysis remain unconsidered. While analyzes of vulnerabilities in existing systems can draw on existing data, this is not an option during early development, as only the targeted function is clear at this phase. At the same time, the final use cases and resulting misuse cases are not.

STRIDE

The name of this threat analysis method is an acronym of the six considered threat classes **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service and **E**levation of privilege that can be used to threaten security objectives [95]. At present, STRIDE is considered to be the state-of-the-art threat analysis method. Each threat class antagonizes at least one security objective. Spoofing a false identity violates the authentication 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 [96].

3.1.3 Security Risk Analysis Methods

In a security risk analysis, the negative impacts and their likelihood are analyzed and evaluated. However, there are serious differences between the established methods used to determine and evaluate the individual factors. The following enumeration describes relevant security risk analysis methods in the context of this work.

CORAS

CORAS [97] is a model-driven IT Security risk assessment method using a graphical notation. The used domain-specific language (called the CORAS language) provides a scheme for graphical-based risk modeling. Furthermore, a method for identification and evaluation of risks as well as an appropriate modeling tool are available. The generated diagrams are analyzed by human analysts, in contrast to the computer-aided approach presented here. The overall risk assessment process consists of eight steps: **(1)** Preparations for the analysis, **(2)** Customer presentation of the target, **(3)** Refining the target descriptions using asset diagrams, **(4)** Approval of target description, **(5)** Risk identification using threat diagrams, **(6)** Risk estimation using threat diagrams, **(7)** Risk evaluation using risk diagrams, **(8)** Risk treatment using treatment diagrams. A typical illustration of this method is the eight footprints shown in Figure 3.3.

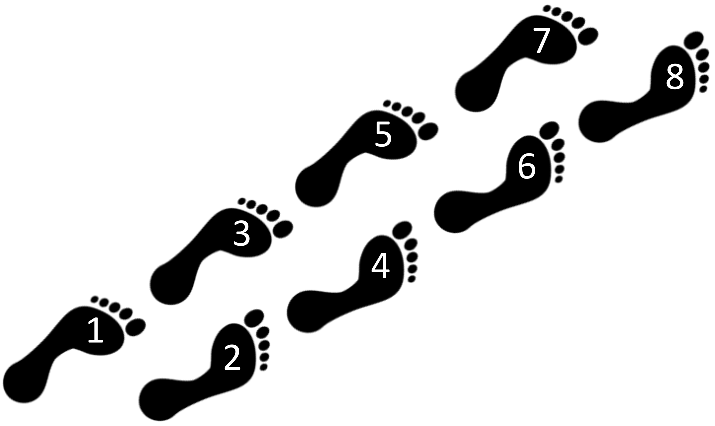


Figure 3.3: The Eight Steps of the CORAS Method [97]

Cyber Security modeling Language (CySeMol)

CySeMol [98] and its extension P^{2}CySeMoL [99] combine UML-based information system modeling with Bayesian attack graphs to assess attack probabilities for a modeled system. The use of the relational model and the thereupon built inference engine allows the evaluation of '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 include iterative dissection, refinement, or a way to model a lack of knowledge about the components of the system.

Factor Analysis of Information Risk (FAIR)

The proprietary FAIR [100] is a framework method for simulation-based information risk analysis. It uses a mathematical simulation model, based on a basic information risk taxonomy. Therefore, it includes a method for measuring the driving risk factors and a computational engine for the mathematical simulation of relationships between the measured 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 reliance on measurable factors, initial holistic 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.

Modular Risk Assessment (MoRA)

As a state-of-the-art method in the automotive domain, Modular Risk Assessment (MoRA) [19, 101, 102] is characterized by a modular structure, supporting a uniform method framework, well-defined work products as interfaces between activities, and different guidelines and catalogs for domain-specific implementation. The need for protection is determined based on individual security objectives. Threats are methodically analyzed and assessed regarding the SuE in a way to allow their interaction to determine a probability level together with countermeasures. Risks result from the combination of possible damage caused by the need for protection and the feasibility of corresponding

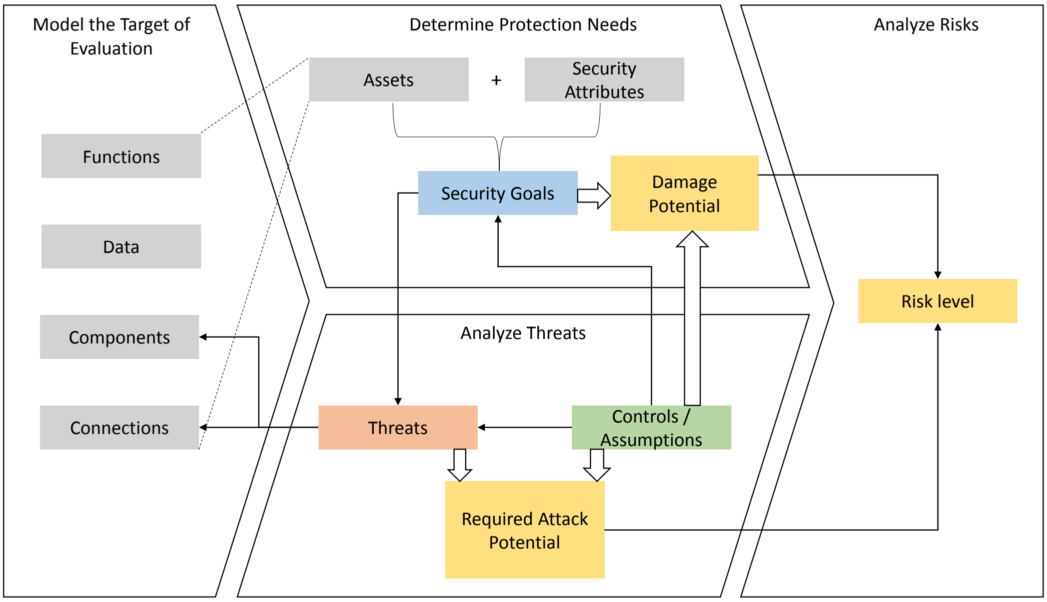


Figure 3.4: Main Activities of the MoRA Risk Assessment Process [19]

threats. Figure 3.4 shows the four core activities of the method framework: *Model the Target of Evaluation*, *Determine Protection Needs*, *Analyze Threats*, and *Analyze Risks*. MoRA relies on an assessment model and a set of catalogs to homogenize assessments within the domain of application. Thus, the assessment model and the catalogs represent a common basic understanding of all stakeholders regarding critical aspects of risk assessment. In the context of this thesis, the main activities of the MoRA process, shown in Figure 3.4, are the framework for the SRA-method described in Chapter 4.

Risk-Tree Based Method for Assessing Risk in Cyber Security (RISKEE)

RISKEE [103] is a practice-oriented, probabilistic risk assessment approach, with strong safety-influences from (S)HARA, FM(V)EA and FAIR. It combines risk calculation with attack trees. Similar to the FAIR approach, the considered factors for the risk calculation are frequency, vulnerability, and magnitude of vulnerabilities. So far, the approach is limited due to bad scalability regarding the available tools [103]. A specialty of the RISKEE approach is the relation and visualization of the calculated and the acceptable risk as a loss-exceeding curve.

Commercial Tools

Recent commercial tools for SRA in the automotive domain are the Yakindu Security Analyst[1], which implements the MoRA method, and the Ansys Medini Analyze[2]. The Microsoft Threat Modeling Tool[3] and fortisee SecuriCAD[4] focus on the identification and assessment of threats to cloud and enterprise IT-environments. Both provide a graphical interface for modeling current and abstract SuE, and assessing their potential security issues. While the Threat Modeling Tool focuses the STRIDE method, described in Section 3.1.2, SecuriCAD also supports simulation-based evaluation of possible attack vectors by Monte Carlo simulation. As dedicated threat assessment tools, both regard coarse-grained attack paths, but lack risk factor propagation or damage transformation.

3.2 Modeling of Security Requirements

Identification, assessment, reuse, and propagation of security requirements are subject to recent research [17, 29, 101, 102, 104]. Requirements are typically expressed in natural language or requirements specification documents, which represent hardly machine-readable and interchangeable notations. Existing definition-languages and formats for general requirements engineering and communication are not tailored to the specific needs of security requirements management in CPS domains.

The Requirements Modeling Language (RML) [105] is based on the idea that a requirements specification should embody a conceptual world model, and that the language for expressing it should provide facilities for organizing and abstracting details, yet at the same time have qualities such as precision, consistency, and unambiguity. Reviewing and evaluating their prior work, the inventors of RML, pointed out that object-oriented representations provide a sound basis for modeling and abstractions, and that refinements are inevitable for modeling large environments [106]. A machine-readable format using such object-oriented representations based on AutomationML for the modeling of network protocols was recently presented by [107].

[1] https://www.itemis.com/de/yakindu/security-analyst/

[2] https://www.ansys.com/products/systems/ansys-medini-analyze-for-cybersecurity

[3] https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[4] https://www.foreseeti.com/securicad/

As a state-of-the-art method for requirements modeling, the Requirements Interchange Format (ReqIF) [108] is supported by the Eclipse Requirements Modeling Framework ProR and IBM DOORS. It defines an interchangeable XML-based format, which can be used to transfer software engineering requirements between different business partners to simplify and streamline the development process. Both approaches, RML and ReqIF, concentrate on intrinsic software engineering requirements to be implemented during development. Accordingly, the format they use is tailored to be interpreted by software developers and not to be incorporated into algorithm-based decisions, as it is required in evolving self-organizing and adapting production systems.

3.3 Network Monitoring and Incident Detection

Network monitoring for incident detection is a security control that can trigger short-term adjustments to the security risk assessment and indicate the need for an appropriate response to sudden changes in the security landscape. A basic taxonomy, classifications as well as reviews on different concepts are provided by surveys on Intrusion Detection Systems (IDS) for Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) [109, 110].

Anomaly detection for CPSs needs to incorporate different data sources and types. Hence, many research projects focus on methods to detect anomalies in specific data types [111, 112, 113, 114, 115]. For a holistic monitoring and detection system however, all data types must be aggregated and monitored. A challenge to this task is the secure transmission of network traces from remote sources with constrained communication capabilities, which is addressed in Section 6.2. The aggregation and analysis are typically done by Security Information and Event Management (SIEM) systems, for which [116] provides an overview of existing systems. Typical approaches for incident detection and classification within ICSs [117, 118, 119] detect local incidents at isolated sites but do not relate the results to remote (previous or present) impairments on a large scale, which is addressed in Section 6.3.

While the network packets are subject to analysis by Network Intrusion Detection Systems (NIDS), logs of regular operation are required to identify malicious patterns [117, 120, 121], and to train algorithms to detect anomalies [117, 122, 123]. Consequently, determining factors for the quality of the classification are the algorithm as well as the amount, quality, and composition of the used training data. A viable source of recorded, labeled, and documented network intrusion test data gathered from enterprise networks are the DARPA challenge datasets from 1998 to 2000 [124]. They contain conventional office IT traffic, including some well-described attacks but no CPS-specific communication. A more recent, CPS-specific Modbus dataset, including attacks and labels, is provided by [125]. Covering the traffic of an industrial test lab for hands-on testing captured during the 4SICS conference in Sweden is provided by [126]. Despite its size, this dataset does not contain labels or information about included attacks. An alternative way to acquire training data are generators [127].

3.4 Incident Classification and Management

In Section 6.3, different concepts for system resilience [128] and incident management [129] are combined to construct a knowledge-graph as a Body of Knowledge (BOK) for current threats, and indicators of compromise. A comprehensive overview of possible network features in ICS that might indicate compromises is provided by [117]. A clear taxonomy and classification of incidents are required for adequate incident response, as well as the exchange of incident information. To this aim, significant efforts by ENISA and the European Commission related to pan-European critical infrastructure protection and incident handling are provided by [130, 131, 132, 133]. Approaches for the behavior-based clustering of malware are presented by [134, 135]. A language and serialization for the exchange of cyber threat intelligence (CTI), among critical infrastructures in a consistent and machine-readable manner, is the Structured Threat Information Expression (STIX) [136]. Consistent relations and classes that support an automated analysis can be received by using machine learning technologies to fit such information into an incident management ontology like CIMBOK [129], the ENISA Ontology [128], or ONTOSEC [90]. Approaches using semantic reasoning on high-level threat information have been presented by [137]. The concept to use a layered architecture of Security Operations Centers at organizational, national, and European levels as the basis for a staged exchange of IoCs was introduced by [138]. Furthermore, the NIST identified several major SCADA-related security aspects [139].

4 Model-based Security Risk Assessment

This chapter addresses the first research question **RQ 1**, for systematic and automatable identification and assessment of security risks for CPSs. The presented approach to address this question is a graph-based security risk assessment method based on a security-specific modeling language. This modeling language supports automated security analysis by descriptions of the System under Evaluation (SuE) in a formal, machine-readable way. Drawing on this modeling, methods for the systematic assessment of potential impacts and threats are presented, whose results are finally merged into a list of assessed risks. In the subsequent Chapter 5, these assessed risks form the basis for the derivation of intrinsic security requirements.

To enable a systematic, automated, model-based security risk assessment, concepts from my following earlier publications are adopted and improved.

The basic system model (cf. Section 4.2) and the impact assessment method (cf. Section 4.3) were published in the paper “Deriving Impact-driven Security Requirements and Monitoring Measures for Industrial IoT” [17], which I presented at the *5th ACM Cyber-Physical System Security Workshop* in 2019. For this thesis, I extended this model, revised the process to propagate impacts along the dependencies, and extended the concept to cover not only the information security triad of confidentiality, integrity, and authenticity but also availability and non-repudiation.

The initial concept of demand-driven model refinement and expansion (cf. Section 4.2.3) was published in the conference paper “Adaptive Modeling for Security Analysis of Networked Control Systems” [23], which I presented at the *4th International Symposium for ICS & SCADA Cyber Security Research* in 2016.

The concept of the graph-based security assessment considering a correlation of threats and mitigations (cf. Sections 4.2.2 and 4.5) was contributed as paper “Modeling

Security Risk Assessments" [19] to the *17th escar Europe – Embedded Security in Cars* in 2019. For this work, the presented risk graph is adapted to the current model.

4.1 Introduction

Protecting CPS against cyberattacks is a challenging task that requires targeted and efficient actions. Awareness of potential risks directly influences the future course of action as it facilitates decisions and helps to avoid unnecessary effort. Since an SRA is simultaneously the cornerstone and the benchmark for a security architecture, it should be conducted as early as possible. Assessments based on checklists are suitable to determine the overall security relevance and to assess simple systems without critical dependencies. While checklists are efficient for simple assessments, detailed assessments of sophisticated systems typically use a formal model to make implicit correlations and dependencies explicit and thus facilitate the overall understanding. The first and already most challenging task in performing an SRA is to understand and model the SuE. In particular, incomplete information on the functionality, connectivity, and parts of the SuE is a challenge often encountered in practice. Nonetheless, an overall and sufficiently detailed model is needed to perform model-based security analysis. While several domain-specific models exist, such as the Smart Grid Architecture Model [140] for the digitized electricity grid, they are suitable for a dependency analysis, but miss SRA-specific relations for relevant information such as security goals, threats, risk mitigations, and uncertainties about specific parts. One approach to improve this situation without the need for entirely new models is to augment the existing domain-specific models with the needed relations and information.

The introduced graph-based process enables the creation of easy-to-build and machine-readable models of the reviewed CPSs, highlighting the relevant assets. Subsequently, such a model enables derivation of abstract intrinsic security requirements in a semi-automatic way (cf. Section 5.3), which is crucial to make complex and highly dynamic critical systems manageable. By formalizing SRAs, best-effort estimates about security risks can be replaced by qualified statements. Where a machine-readable model is available, several parts of the threat analysis can be automated by using well-established approaches such as STRIDE or ATs.

4.2 System Model for Security Risk Assessment

> "All models are wrong, but some are useful" G. Box, 1979 [141]

This section addresses the generation of useful CPS models for assessing security risks and deriving requirements.

4.2.1 Basic System Model

The creation of a useful model requires a suitable modeling language. Given the dynamic communication behavior of CPSs, assets and dependencies are subject to spontaneous change, which means that the virtualized model of a SuE must be easy to maintain. Often, the required information is incomplete or inconsistent, which makes model creation a major challenge. Some of the security risk assessment methods introduced in Section 3.1.3 provide modeling languages that enable highly expressive models. In practice, however, it turned out that complex modeling often causes an inhibition threshold by complicating the modeling process. A well-proven way to unify and relate heterogeneous information is a formal model with an appropriate ontology, including class-hierarchies. The presented Security-specific Modeling Language (SSML) consists of a small set of base classes and relations, as defined in Listing 4.1 on page 47 and shown in Figure 4.1 on the next page. For the sake of brevity, the listings in this work do not include individual namespaces.

Information typically used for threat modeling are data-flow, control-flow, and sequence diagrams, as well as network plans, inventory lists, and use-case scenarios. Due to domain-specific description differences, the acquisition of further information can be a real challenge, requiring some mutual understanding between application experts and security analysts. Reaching such an understanding is crucial to comprehend, reuse, and enhance existing domain-specific models with the needed information.

Targeted augmentation of hierarchical refinements and implementation details enables downstream applications, such as requirements derivation and automated incident assessment.

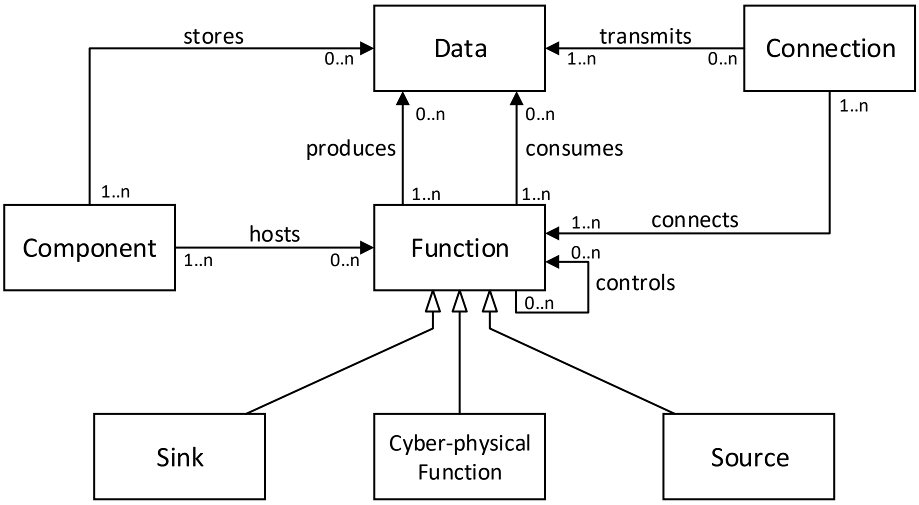


Figure 4.1: Meta-model of the Security-specific Modeling Language

Classes

The following base classes are introduced for a machine-readable modeling of a SuE:

Function: An operation that consumes an input to produce some output. It can be a sophisticated data app but also a simple media converter. A function consuming input from outside the model to produce data is called a `Source`, while a function only consuming data from the model to act outside is a `Sink`. A third special type is a cyber-physical function (`CPF`), which consumes and produces data within the model but also operates outside of it.

Component: A (physical or virtual) instance that hosts functions or stores data. It may consist of several sub-components.

Connection: Enables functions to exchange data with each other. This includes inter-process-communication within the same component, as well as communication to distant components via networks. Where necessary, each connection can in turn be iteratively refined to a function between two new connections in order to model several communication jumps.

Data: Any type of information that can be consumed or produced by a function, stored at a component, or transmitted via a connection.

The declaration of these classes is shown in lines 2–9 of Listing 4.1.

Listing 4.1: Definition of Classes and Relations of the Basic System Model

```
 1 # Classes
 2 Asset       type        Class .
 3 Function    subClassOf  Asset .
 4 Component   subClassOf  Asset .
 5 Connection  subClassOf  Asset .
 6 Data        subClassOf  Asset .
 7 Sink        subClassOf  Function .
 8 Source      subClassOf  Function .
 9 CPF         subClassOf  Function .
10
11 # Relations
12 hosts     a  ObjectProperty ; domain Component  ; range Function .
13 connects  a  ObjectProperty ; domain Connection ; range Function .
14 controls  a  ObjectProperty ; domain Function   ; range Function .
15 stores    a  ObjectProperty ; domain Component  ; range Data .
16 transmits a  ObjectProperty ; domain Connection ; range Data .
17 produces  a  ObjectProperty ; domain Function   ; range Data .
18 consumes  a  ObjectProperty ; domain Function   ; range Data .
```

Relations

Linking the introduced classes with each other is facilitated by the following relations. Their declaration is shown in lines 12–18 of Listing 4.1.

hosts relates a Function to the Component by which it is executed.

connects describes by which Connection a Function can communicate.

controls enables logical Connections between two Functions in terms that an attacker can use a controlling Function (A) to manipulate a controlled Function (B). This way, the attacker does not need to compromise the Connection between these Functions or the Component hosting the controlled Function B.

stores indicates at which Component a Data is stored.

transmits describes by which Connection a Data is transmitted.

produces indicates the Function sending a Data.

consumes indicates the Function receiving a Data.

4.2.2 Extension for Security Management

By augmenting the system model with security-relevant information, including security goals, threats, and threat mitigations, it becomes usable for the assessment and management of security risks.

To this aim, the following security-specific classes and relations are introduced, as shown in Figure 4.2 on the next page and Listing 4.2 on page 50.

SecGoal Security goals are described as elements of the class `SecGoal`. The objects of this class are generated for each combination of relevant security objective and basic system model, i.e. function, component, connection, and data, to which they are related by an *isAssetOf* relation. They hold the individual Cybersecurity Impact Level, and the Required Attack Potential, derived by the impact and threat assessments.

Threat Derived during threat assessment, instances of this class hold the attacker type specific Provided Attack Potential for each combination of threat and basic system model class. They are assigned to the relevant security goals by a *threatens* relation.

SecMitigation Threat Mitigations encompass the three sub-classes, security controls (`SecControl`), assumptions (`SecAssumption`) and measures (`SecMeasure`) (cf. Section 2.3.2 on page 23). Elements of the `SecMitigation` class are assigned to `Threats` they mitigate by a *mitigates* relation. In contrast to the other two subclasses, `SecControl` objects have a further *dependsOn* relation to the `SecGoal` of the function and data that are required for proper operation of the protection. Based in their influence on the mitigated threats, they hold a Δ of the Required Attack Potential (RAP) or Cybersecurity Impact Level (CIL).

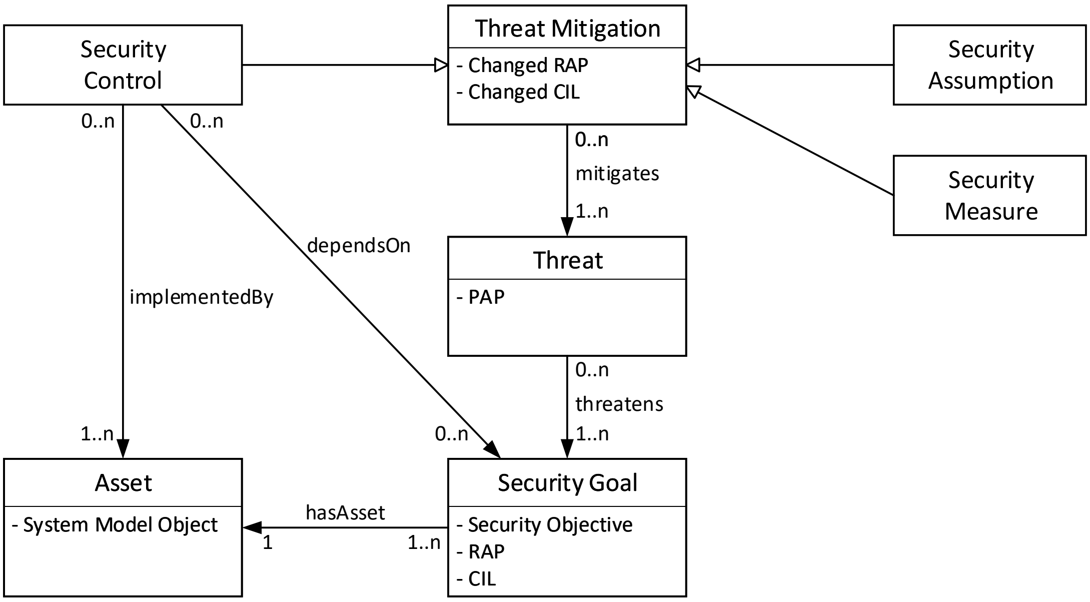


Figure 4.2: Extensions for Security Modeling

4.2.3 Systematic Refinement and Namespaces

Combining hierarchical refinement and namespaces enables a progressive modeling. An optimal way to make sure all aspects of a SuE are covered is to model it during the design stage and to continuously refine it during ongoing development. However, a model can be created at any phase of the system lifecycle.

Initially, the SuE is regarded as a monolithic component with a single function and connections to the outside world. As soon as implementation details become known, a first refinement round is started by decomposing the functions and adding individual connections. Functions, connections, components, and data are iteratively extended to a refined model. After their decomposition, functions can be relocated from the initial monolithic component to newly refined sub-components. Each refinement round thereby provides a higher level of detail.

In addition to the structural expansion of entities, sub-concepts of the used namespace provide further implicit information. The purpose of these sub-concepts is twofold: first, the implementation-specific knowledge required for an in-depth security analysis can be stored implicitly for the specified subtypes. Second, the recognition and reuse of previously refined structures become possible. Where no implementation information

Listing 4.2: Definition of Security relevant Subclasses and Relations

```
# Classes
Threat         a          Class      .
SecGoal        a          Class      .
Mitigation     a          Class      .
SecControl     subClassOf  Mitigation .
SecAssumption  subClassOf  Mitigation .
SecMeasure     subClassOf  Mitigation .

# Relations
dependsOn     a ObjectProperty ; domain SecControl ; range  SecGoal   .
hasAsset      a ObjectProperty ; domain SecGoal    ; range  Asset     .
implementedBy a ObjectProperty ; domain SecGoal    ; range SecControl .
mitigates     a ObjectProperty ; domain Mitigation ; range Threat     .
threatens     a ObjectProperty ; domain Threat     ; range SecGoal    .
```

is available, the according super-class has to be used to model an instance. This often happens due to incomplete documentation or uncooperative vendors. Occasionally, the required knowledge can still be obtained by reasoning or reverse engineering, which is a clearly preferable solution.

To avoid deviations from the practical implementations, it is reasonable to follow the implementation hierarchy and keep parts together. To this purpose, the introduction of sub-classes as parts of individualized namespaces significantly improves the clarity of descriptions and makes the modeling process more intuitive. As an example, a layer of sub-classes for functions might be firmware, operating system, and application. Implementation information can be used to reasonably assess threat vectors and exclude weaknesses and vulnerabilities that can reliably be denied (e.g., the embedded firmware of a Programmable Logic Controller (PLC) is not affected by a vulnerability in a printer spooler). This approach further allows a model to be fine-grained at sensitive areas and made coarser where only limited information is available or required. Sensitive areas are typically those with a high cybersecurity impact (cf. Section 4.3). While the impact assessment presented in the following Section 4.3 does not consider the implementation details, they are essential for the threat assessment in Section 4.4 and the derivation of intrinsic security requirements (cf. Section 5.3). A detailed description of the successive refinement steps is presented and demonstrated in [23].

While refinement to a basic level of granularity is necessary for a certain expressiveness of the analysis, it often occurs that components and functions cannot or need not to be refined beyond a certain level. These items are then considered black boxes. Typical reasons for missing details are uncertainty during development, and the unwillingness of suppliers to provide information about their products.

Refinement of the abstract system model towards a detailed model of the implemented system, including resource identifiers for product and version information, enables automated references to security-specific databases (cf. Section 2.2).

A major advantage of semantic modeling is that in addition to the explicitly gathered information, further correlations can be expressed by axioms. Even in large systems, where it would be difficult to keep track of consequences if done manually, an automated reasoner can reveal inconsistencies and evaluate models using queries.

Due to uncertainty about the specifics that might enable the exclusion of potential risks, usage of high-level descriptions leads to worst-case assumptions during assessments, as all potential vulnerabilities for that instance's super-class must be considered.

4.2.4 Modeling Systems for Analysis

The first step of creating a valid system model is the acquisition phase, during which the necessary information is gathered. Depending on the information available, there are different ways of doing this. For well-described systems, it may be sufficient to merge the information from existing diagrams. Interviews with architects, experts, and operators are often necessary to achieve the desired results. Occasionally, an automated inventory can also be carried out for running systems using network scanners such as Nessus.

The acquisition phase is succeeded by the modeling phase. Here, the analyst starts with the creation of a basic system model of the SuE. This model is later augmented with the relevant security information regarding impacts for the security objectives. Clustering the static and dynamic instances during modeling is recommended as it improves maintainability. In many cases, it is necessary to refine the model to increase information about the system. Avoiding unnecessary effort and preferring a simple, maintainable model, a detailed specification of the instances can be added later where necessary. This expansion needs not to be done evenly; that is, some parts of the system can be described more in detail than others. An exemplary model of an SuE is shown in Listing 7.1 on page 104 as part of the Use Case Evaluation in Chapter 7.

4.3 Impact Assessment

To devise the impact of a potential risk by violated security goals, it is necessary to assess their worth, meaning that aspects such as injuries, financial losses, and legal infringements must be put in relation to each other. To this purpose, the security objectives of each instance are rated regarding their cybersecurity impact in a Cybersecurity Impact Level (CIL). While the levels can be defined for each organization individually, a scale ranging from level 0, no relevance, to level 4, very high relevance, as suggested in IEC 62443-4-2 [14] has proven to be well suited. The individual impact levels are assigned based on potential damage regardless of its cause. As a starting point, it is necessary to provide some initial ratings from historical values, expert opinions, ground truth, or similar, in order to systematically propagate them along the model. In the following, a description of the individual impact levels if provided.

Confidentiality Impact Level ($\mathrm{CIL}_{\mathrm{Conf}}$): The impact of cases in which a confidential information is disclosed. This includes know-how, intellectual property, but also cryptographic keys.

Integrity Impact Level ($\mathrm{CIL}_{\mathrm{Int}}$): The impact of cases in which the integrity of the instance is compromised. This can be an altered component or function, as well as a manipulated connection or data.

Availability Impact Level ($\mathrm{CIL}_{\mathrm{Ava}}$): The impact of cases in which the instance is unavailable. This can be a broken component, function, connection, or unavailable data.

Authenticity Impact Level ($\mathrm{CIL}_{\mathrm{Aut}}$): The impact of cases in which the authenticity of the instance is compromised. The relevance of $\mathrm{CIL}_{\mathrm{Aut}}$ depends on the nature of the overall system. A typical discussion that should be initially clarified is whether inauthentic instances can be valid data in terms of integrity. Usually, fake data such as spoofed instructions or manipulated values are relevant here. Besides, counterfeit components can cause damage in certain application environments.

Non-Repudiation Impact Level ($\mathrm{CIL}_{\mathrm{NR}}$): The impact of cases in which one party can successfully dispute to be the sender or editor of a data. The relevance of $\mathrm{CIL}_{\mathrm{NR}}$ depends on the nature of the overall system. It is particularly important for orders on virtual marketplaces and affects functions, data, and connections.

4.3.1 Impact Propagation

Looking at the dependency tree, CILs can be propagated up- and downwards through the model. Guides on creating such trees are provided by [29] and [83]. With the following propagation method, the CIL of all instances of the models is calculated iteratively. Note that only a small set of initial data D_0 (for confidentiality) and sink-functions F_0 (for all other security objectives) must be rated manually in order to propagate the CIL through the remaining model. To cope with special cases, further manual assessments before or after propagation can also be reasonable. The process to evaluate the impact of the security objectives integrity, availability, authenticity, and non-repudiation for all instances, propagates upwards from the sink to the source. The only exception of this direction is the evaluation of the confidentiality-specific CIL_{Conf}, which requires a manual rating of the sources D_0 and propagation down to the sinks. An exemplary application of this process is shown in Section 7.3.1 as part of the use case evaluation in Chapter 7.

Top-Down Propagation Based on the initial assessed CIL_{Conf} of a source and special instances, the confidentiality levels are propagated top-down from their initial manually rated source data. First, the CIL_{Conf} of a function is evaluated to be the maximum CIL_{Conf} of the source data D_0 that are consumed by this function and, if applicable, an intrinsic value (e.g., a valuable sophisticated algorithm). Second, the CIL_{Conf} of a connection is evaluated to be the maximum CIL_{Conf} of data that are transferred through this connection. Third, the CIL_{Conf} of a component is evaluated to be the maximum CIL_{Conf} of functions that are hosted by this component. Fourth, the CIL_{Conf} of previously unrated data is evaluated to be the CIL_{Conf} of connections they are transmitted through, functions they are produced by, or components they are stored at, respectively.

Bottom-Up Propagation To determine the levels for the other security objectives, the bottom-up propagating objective relevance levels CIL_{Ava}, CIL_{Int}, CIL_{Aut}, and CIL_{NR}, are propagated from their initial manually rated sink functions F_0. Be CIL_X the rating for a security objective X from this enumeration. First, the CIL_X of the data consumed by sink function F_0 is evaluated to be the maximum CIL_X of the function that consumes this data. Second, the CIL_X of a connection is evaluated to be the maximum CIL_X of data that are transferred through this connection. Third, the CIL_X of a component is evaluated to be the maximum CIL_X of functions that are hosted by this component. Fourth, the CIL_X of previously unrated functions is evaluated to be the level of data they produce.

Cross-class Inheritance for Security Measures A special case of inheritance applies to security measures that inherit the level of the instance they protect. Thereby, the class of the inherited security objective changes while its level remains, typically towards integrity or availability of the security control (function). An example of such a security control is a sanitizing function, which lowers the confidentiality level of a data by removing sensitive information. Thereby the integrity of the function inherits the confidentiality level of the unsanitized data for its own integrity, while the sanitized data can have a lower $\mathrm{CIL}_{\mathrm{Conf}}$. Such relationships must be described in advance of further propagation.

4.3.2 Instance-specific Impact Aggregation

The output of the propagation process is a list of assessed security objectives, indicating the potential impact on violating the associated security goals. The aggregation of these levels per instance leads to an overall rating, comparable to the Safety Integrity Level (SIL) [142]. It creates comparability between impairments and thus indicates the appropriate attention and severity that should be given to the instance. In order to aggregate the overall effect of an instance, various situation-specific approaches can be applied. Intuitive approaches are to determine the maximum of its intrinsic CIL, as shown in Equation 4.1, to use a discrete aggregation matrix, or a classification scheme similar to the Automotive Safety Integrity Levels (ASIL) [143].

$$\mathrm{Impact} = \max(\mathrm{CIL}_{\mathrm{Conf}}, \mathrm{CIL}_{\mathrm{Ava}}, \mathrm{CIL}_{\mathrm{Int}}, \mathrm{CIL}_{\mathrm{Aut}}, \mathrm{CIL}_{\mathrm{NR}}) \tag{4.1}$$

4.4 Threat Assessment

Assessing threats after the impact assessment can be advantageous, when violations of security goals without consequences may be omitted. Depending on the type of access, a threat may directly threaten a specific security goal, or require a series of previously successful threats as an attack vector, as described by [19]. For each threat and attack vector, the RAP and threat factors must be determined. A potential basis for this, based on the [41] is show in Table 4.1. The used deviations from the standard are printed in italics. While the requirements and their values should be adapted individually, the threat factors and the differentiated levels can often be adopted. This determination can be either done manually, using sources like expert knowledge, historical data, and ground truth, or automatically, based on a threat assessment catalog.

While utilizing such a catalog requires standardized implementations and sufficient granularity of the model, it allows for a consistent, unbiased assessment and a high degree of automation. An exemplary application of this process is shown in Section 7.3.2 as part of the use case evaluation in Chapter 7.

Table 4.1: Rated Threat Factors for the Attack Potential Calculation, based on [41]

Threat Factor	Requirement	Value
Knowledge of the TOE	Public	0
	Restricted	3
	Sensitive	7
	Critical	11
Equipment	Standard	0
	Specialized	4
	Bespoke	7
	Multiple bespoke	9
Window of Opportunity	Unnecessary / unlimited	0
	Remote (added)	*1*
	Easy (adjusted)	*1->3*
	Moderate (adjusted)	*4->6*
	Difficult	10
	None	-

Threat Factor	Requirement	Value
Elapsed Time	≤ one day	0
	≤ one week	1
	≤ two weeks	2
	≤ one month	4
	≤ two months	7
	≤ three months	10
	≤ four months	13
	≤ five months	15
	≤ six months	17
	> six months	19
Expertise	Layman	0
	Proficient	3
	Expert	6
	Multiple experts	8

4.4.1 Threat Elicitation

A simple and effective way to elicitate all potential threats is to map the base model instance classes to their relevant threat classes, as shown in Table 4.2. Depending on whether the asset is a data, a component, a function, or a connection, only certain threat classes apply.

Table 4.2: Threatened Security Objectives by Threat- and Instance-Classes

Class	**Spoofing**	**Tampering**	**Repudiation**	**Information Disclosure**	**Denial of Service**	**Elevation of Privilege**
Function	Integrity Authenticity	Confidentiality Integrity Availability Authenticity	Non-repudiation	Confidentiality	Availability	Integrity Authenticity
Component						Integrity
Data						
Connection						

4.4.2 Estimation of Required and Provided Attack Potentials

For the ability to successfully execute a threat, the Provided Attack Potential (PAP) of an attacker must equal or exceed the Required Attack Potential (RAP). The estimation of this RAP is done in a standardized and normalized form, based on an agreed set of categories, as shown in Equation 4.2.

$$\text{AP} = [\text{Time}|\text{Expertise}|\text{Knowledge}|\text{Type of Access}|\text{Equipment}] \tag{4.2}$$

Common factors for the characterization of attack potential are the elapsed time, specialist expertise, knowledge of the SuE, window of opportunity, and equipment required for exploitation [41]. Here, each of the factors are vectors, including their ratings as depicted in Table 4.1. Examples for such threat specific estimations are shown in Table 4.3 for the RAP, and in Table 4.4 for the attacker type specific Provided Attack Potential (PAP). Depending on the individual circumstances, further factors and refinements can be necessary, such as the differentiation of the type of access.

Threats can often be aggregated, e.g., when multiple data elements are stored in the same component, transmitted over the same connection, or identical technologies

Table 4.3: Exemplary Estimated Required Attack Potential per Threat

Identifier	Expertise	Knowledge	Equipment	Time	Type of Access
Threat 1	Layman	Public	Standard	Days	Easy
Threat 2	Proficient	Restricted	Specialized	Weeks	Moderate
Threat 3	Expert	Sensitive	Bespoke	Decades	Remote
Threat 4	Multiple experts	Critical	Specialized	Weeks	Easy

Table 4.4: Exemplary Estimation of the Provided Attack Potential per Attacker Type

Attacker Type	Expertise	Knowledge	Equipment	Time	Type of Access
Script Kiddy	Layman	Public	Standard	Days	Remote
Hackivist	Proficient	Public	Specialized	Weeks	Remote
Organized criminal	Expert	Sensitive	Bespoke	Months	Remote
Intelligence service	Multiple experts	Critical	Multiple bespoke	Years	Difficult

are used. An important question, thereby, is if each factor must be provided for each execution of the threat or only once, e.g., to extract a master key [2].

Vulnerabilities can significantly reduce the RAP by lowering threat factors. A vulnerability caused by a faulty Hardware Security Module (HSM) can, for example, greatly reduce the entropy and thus the time required for an attack from years to days [21]. Mitigations, in contrast either reduce the CILs, or increase the RAP by increasing a threat factor, such as adding the need for bespoke equipment.

A common way to avoid ambiguities is to classify and evaluate the RAP by a score system, based on a discrete scale, and the possible types of access to the SuE. The estimated probability of a successful attack is then determined from the calculated effort. The result of the assessment is an estimate of the potential that an attacker is likely to expend to achieve his objective in violating a security goal.

In summary, the RAP can be regarded as the capability an attacker has to possess to execute a threat.

4.4.3 Attacker Motivation - Costs and Benefits

While the ability to execute an attack is determined by the RAP and PAP, the attractiveness may be assessed by the benefits and costs from an attacker's perspective. The expected benefits usually define the effort an attacker is willing to expend. In contrast to estimating the RAP, which is possible with a certain degree of uncertainty, estimating the cost-benefit-ratio is rarely more than a vague suspicion and omitted by most SRA methods. While the ratio cannot be used to reliably nullify a risk, it may highlight extraordinary cheap or expensive attacks.

A basis for this analysis is the Equation 4.3 for the choice to commit a crime [57], previously described in Section 2.3.1 on page 20.

$$\underbrace{M_{\mathrm{b}} + P_{\mathrm{b}}}_{\text{Benefits}} > \underbrace{O_{\mathrm{cm}} + O_{\mathrm{cp}} \cdot P_{\mathrm{a}} \cdot P_{\mathrm{c}}}_{\text{Costs}} \tag{4.3}$$

As pointed out, estimating the psychic benefits P_{b}, psychic opportunity costs O_{cp}, the probabilities of apprehension P_{a} and conviction P_{c} is, in contrast to the monetary benefits M_{b} and costs O_{cm} rarely possible in a meaningful way. In such cases where a qualified assumption about the costs and benefits is possible, they can be formulated in the same categories as the required attack potential and directly compared.

4.4.4 Attack Vector Identification and Assessment

Similar to the dependency on other security goals, attacks can either consist of a threat targeting a singular security goal, or attack vectors consisting of a series of threats. A standard approach to identify attack vectors is to create an individual attack tree for each attack goal.

The attack vector identification starts with the identification of each potentially targeted security goal. Approaches can be differentiated according to the potential point of entry by which an attacker reaches or accesses his target system. While a tuner has physical access to the components of, e.g., a car, tampering their functionality in terms of software is often protected by restrictive security controls, such as Access-Control Lists (ACLs) and HSMs. Where the attacker has no direct access to his target, he must perform a multi-stage attack. Dependency trees, indicating the potential attack vectors for each relevant security goal, can be derived by systematical querying the system model [29]. Each traverse of a node and edge in such a tree may require specific attack factors and potentials. The resulting graph is then analyzed using methods of graph theory for directed graphs, taking into account the required capabilities and in place threat mitigations. To avoid inconsistencies, this graph should always be freshly generated from the current model. Furthermore, a regression analysis is performed after each iteration of the system model.

While an abstract model leads to generic results in the development phase, a more specific picture of the security situation can be derived from detailed implementation information. Predicting the motivation or precise movement pattern of an attacker is hardly possible. A common way is to assume that he chooses the "easiest path", which is rarely known to the attacker. Therefore, it is not unusual that a less probable, often less monitored vector is chosen.

4.4.5 Probability Estimation

A threat must be considered possible, whenever the PAP of an attacker matches the RAP. If further the benefits outweigh the costs, as described by Equation 4.3, it is considered attractive. To determine a value for the probability, the attractiveness and the possibility are related as captured in Equation 4.4.

$$\text{Probability} = \begin{cases} 1 - \frac{\text{RAP}}{\text{PAP}} \cdot \frac{\text{Costs}}{\text{Benefits}} & \text{, if RAP} \leq \text{PAP and Costs} \leq \text{Benefits} \\ 0 & \text{, if RAP} > \text{PAP} \quad \text{or} \quad \text{Costs} > \text{Benefits} \end{cases} \tag{4.4}$$

This probability might be shifted by considering mitigations increasing the required effort, and by vulnerabilities reducing it. While the material costs might be high for the initial development of tools and methods, the repetition of many attacks may be easy, especially when no expensive or bespoke equipment is required. Methods to recognize and manage such situations are subject of Chapter 6.

4.5 Risk Determination and Aggregation

The individual risk rating for each security goal can be determined as shown in Equation 4.5, as product of the CIL and the probability value from Equation 4.4.

$$\text{Risk} = \text{Impact} \cdot \text{Probability} \tag{4.5}$$

Where applicable, these risk ratings may be affected by threat mitigations and vulnerabilities that shift the required attack potential, or decrease the impact, as shown in Figure 4.3. This process is demonstrated in Section 7.3.3, as part of the use case evaluation in Chapter 7. The derivation of technical security measures and the management of security requirements, in general, are subject of the following Chapter 5.

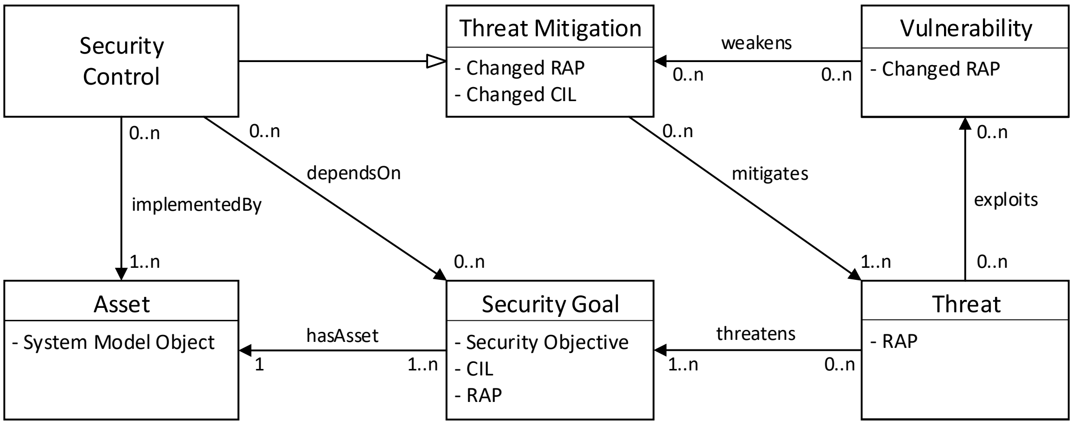


Figure 4.3: The Security Risk Graph

For a better comprehensibility, the identified security goal specific risks can be aggregated into damage scenarios [16]. These scenarios combine all security goal violations that result to the identical consequences. The probability of each damage scenario is inherited from the security goal with the highest probability resulting in this scenario.

4.6 Discussion

This chapter presented a method for the systematic assessment of security-risks. The basis for this analysis is an introduced basic, machine-readable model that stores the necessary information. For the systematic evaluation of potential damage resulting from the violation of security objectives, these are assessed concerning their effects and propagated by the model according to their dependencies. The likelihood depends on the attractiveness of the attack for the considered attacker models and the comparison of expected and required attack potential.

Spontaneous risk assessments are part of everyday life. This often simple task becomes difficult at the latest when it comes to providing well-founded assessments of complex subjects that must be comprehensible to others and can be used by them as a basis for strategic decisions. The presented model-based security risk assessment process prevents misjudgments through normalized criteria and systematic procedures. Impact and threat assessments are conducted independently of each other on the common basis of a jointly developed and agreed model. The combination of both assessments as qualified factors leads to a systematic SRA.

The presented modeling method is a tool to describe a SuE properly for security analysis in a machine-readable way. Although several alternative modeling approaches are also valid, the presented one proved to be intuitive, manageable, and capable. While the modeling of data and component objects is common to most of the related SRA-approaches (cf. Section 3.1.3), the explicit modeling and annotation of security properties to function and connection objects are subject to continuous discussion. The implementation of functions is often considered as (sub-)components, or directly differentiated into more detailed sub-classes. Modeling connections as threatenable instances, in contrast to the alternative of annotations to functions or as (directed) interfaces, enables them to have individual object-properties. This makes connection objects suitable both for describing the inter-process communication between two processes on the same ECU as well as for a remote connection across several remote networks. Furthermore, the *produces*, *consumes*, and *transmits* relations simultaneously enable the modeling of individual end-to-end communication and sender-subscriber models on shared or redundant communication media.

This modeling supports consideration of attacks that are otherwise difficult to integrate, such as the addition of an interface or function by an attacker. The first case is considered a manipulation of the producing function, the second of the hosting component.

For the impact assessment, a common understanding between domain and security experts can usually be achieved in a short time. In contrast, threat assessment and its discussion with function or component managers are often challenging tasks because threats introduce a dynamic that they are not yet used to from other areas. When assessing threats, it is examined separately whether an attacker is able to implement them and whether the execution could be attractive for him. In most cases, the assessment of attractiveness can be treated as an attacker model specific binary decision. If an attacker might benefit from executing a specific threat, assessing the attractiveness is particularly valuable for the prioritization of security measures.

The aggregation of security goal specific risks into damage scenarios at risk provides a comprehensible representation of the overall situation. However, a security goal specific granularity is required for mitigation and requirements derivation. The sole number of security goal specific risks gives an impression on the overall complexity of risk assessment, but not necessarily of the SuE due to frequent overlap.

While adequate consideration of the numerous impact and threat factors pushes manual analyses to the limits of their complexity, the systematic, iterative process can be mostly automated. Catalogs and object-orientation further enable the reuse of elements once analyzed and evaluated.

Risk scaling and accumulation often differ significantly depending on the domain and business model. Considering all threat mitigations, the number of residual risks should usually be negligible, especially for safety risks. The derivation and management of such threat mitigation are subject of the following Chapter 5.

Apart from operational systems whose shutdown leads to higher losses, severe residual risks are usually a reason to stop the use or further development of a system until the situation can be resolved. Dealing with the dynamics of evolving threats requires the manufacturers and operators of CPSs to develop new processes and ways of thinking, to keep their products at a continuous level of security, which is a subject of Chapter 6. A use case example for the successive execution of the introduced modeling and security risk assessments processes is provided at Section 7.3.

Knowledge of the risks indicates the need for action and enables targeted measures to be demanded and their effects to be anticipated. The formalization, derivation and coordination of such measures is the subject of the following chapter.

5 Derivation and Management of Security Requirements

This chapter addresses the second and third research questions. **RQ 2**, on formulating security requirements to support the achievement of the indicated level of protection, is approached by utilizing the work product of the previously described security risk assessment to derive and prioritize intrinsic security requirements. For **RQ 3**, on matching extrinsic security requirements demanded by external entities with the implemented capabilities, an approach for their management by means of formalization and automated fulfillment validation is presented.

To this purpose, concepts from my following earlier publications are adopted and improved.

The process to derive intrinsic security requirements (cf. Section 5.3) from a model-based impact assessment was published in the paper "Deriving Impact-driven Security Requirements and Monitoring Measures for Industrial IoT" [17], which I presented at the *5th ACM Cyber-Physical System Security Workshop* in 2019.

The methods for the formalization, hierarchic structuring, and communication of security requirements (cf. Section 5.2), as well as matchmaking with the capabilities of the existing system (cf. Section 5.4), were published at the *24th IEEE Conference on Emerging Technologies and Factory Automation* in 2019, as paper "A Unified Architecture for Industrial IoT Security Requirements in Open Platform Communications" [20].

5.1 Introduction

> “Real threats violate requirements.”, Adam Shostack, 2014 [96]

Knowledge of the potential risks alone is necessary, but not sufficient. As secure operation requires reliable operation within the expected boundaries, requests for effective threat mitigations must be made and implemented.

The therefore used security requirements must be precisely formulated, prioritized, and validated, which is subject of Section 5.2. Two general types of security requirements are to be distinguished. Intrinsic security requirements are indicated by an SRA, as described in the previous Chapter 4, and are subject of Section 5.3. Extrinsic security requirements originate from external entities, such as regulators, customers, and suppliers. Their management is subject of Section 5.4. An evaluation of the Security Requirements Data Model is provided in Section 5.2.4, followed by a discussion of the exposed methods in Section 5.5.

A common problem for the management of security requirements are different languages, formats, and granularity used for their specification. To enable automatic validation, whether an implemented capability meets a new security requirement, they must become comparable. Besides misunderstandings caused by foreign languages, the various presentations, such as user stories, use case diagrams, and spreadsheets, significantly complicate an automated validation. In particular, free-text-based descriptions lack the ability to be machine-readable but require additional, error-prone natural language processing. A compromise are human-readable, list-based formats, which allow the comparison between demanded and implemented requirements by itemizing them. Assuming that everyone uses the same wording for the same requirements, these would also be machine-readable. However, if implementation details of a requirement, such as the minimum length of an RSA key, are embedded in sentences, significant processing is required to retrieve this information for further automated processing. In a table-based format, these parameters can be listed in additional columns yielding a more machine-readable representation for automation while still being human-readable. In contrast, an object-oriented security requirements data model is only human-readable An effective method that supports a uniform level of security is the use of patterns that map appropriate threat mitigations from a catalog to each security objective at risk, provided a suitable visualization application exists. On the other hand, the format is fixed and does not contain natural language's variance. This allows for much easier comparison between the requirements of different companies or with implementation descriptions.

Formulating security requirements and implemented capabilities in a unified language yields the ability to compare them against each other for business ICT systems down to embedded devices. The subsequently presented machine-readable security requirements data model enables the fully automatable communication of security requirement along a supply chain. Thus, it allows for higher and faster automation in the heavily heterogeneous landscape of CPSs in the field.

Developing an effective security concept for a specific, sensitive CPS or parts of it is a responsible and challenging task. Automating the process of deriving intrinsic security requirements from the identified threats and prioritizing them according to their individual risk can significantly support this task. An effective method that supports a consistent level of security is the use of patterns that map appropriate threat mitigations from a catalog to each security objective at risk. Thereby, all potential mitigations are initially presented but then pruned according to their impact and effect. Monitoring the compliance and effectiveness of implemented requirements, as well as the proper response to incidents, are the subject of Chapter 6.

5.2 Processable Security Requirements

Data models provide several ways to manage security requirements. Common practice for communicating security requirements relies on textual forms that are hardly machine-readable. Internationally acknowledged certification methods like the *Common Criteria for Information Technology Security Evaluation* [41] use checklists and spreadsheets to represent their requirements. For the widespread text-based forms, the primary advantage is their human readability. However, they lack the possibility to be machine-readable and cannot easily be compared. A list-based format enables a basic comparison of demanded and implemented security requirements by both machines and humans. This can be further improved by an object-oriented data model, including hierarchical categories, and attribute sets.

Table 5.1: Feature Comparison of Requirement Enumeration Techniques

Feature	running text	lists	tables	object-oriented
human readability	✓	✓	✓	(✓)
comparability		(✓)	(✓)	✓
machine readability		(✓)	✓	✓

5.2.1 Object-Orientation

An object-oriented model can be used to automatically derive a hierarchical representation of security requirements. To transform the enumerations of security requirements to such an object-oriented data model, a set of requirement classes is developed, which describes the possible security requirements space. Each requirement then becomes an instance of such a class, whereas the definition of attributes contains required implementation details for the requirement. The inheritance in the object-oriented representation allows incorporating abstract or vague requirements. Such an abstract requirement can later be augmented with implementation details. Therefore, the data model is extendable by the introduction of new subclasses. This definition of a class hierarchy automatically leads to a hierarchy of security requirements, which enables the comparison of vague requirements with precise implementation details. By referencing instances of other requirement classes, it is also possible to relate supporting or implied requirements. As the object-oriented design is a widespread paradigm, the model can be used at any phase of the CPS-lifecycle.

5.2.2 Categorization

To develop a unified data model for the different types of security requirements they are clustered into high level categories. Some security requirements describe the internals of the systems, like hard drive encryption for confidentiality or load balancing for enhanced availability. Further, a central point of security requirements is the implementation of interfaces. However, several requirements cannot be classified with these rather technical, internal or communication, functional groups. Many of the requirements consider the way updates to software or hardware should be treated and how the product itself should be developed, e.g., if penetration testing or audits must be carried out during

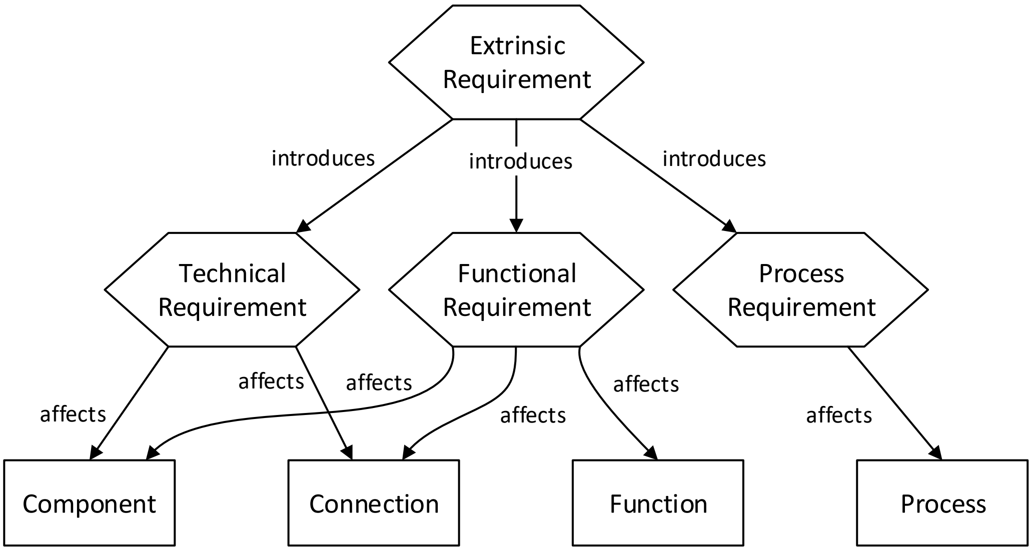


Figure 5.1: Relations for Extrinsic Requirements

the development phase. These types of requirements refer to the manufacturing process of the product.

Therefore, three main categories are defined: Technical (Non-functional), functional, and process security requirements.

Technical Security Requirements directly refer to device inherent, technical properties, the internal processing, or the system itself. In contrast to operation specific functional- and non-function requirements, they are mostly transparent to the outside during operation. Figure 5.2 shows a hierarchy of requirement classes within this category. Each node in the tree resembles a class of security requirements, whereas child nodes refer to subclasses in the object-oriented representation. As indicated in the tree, each abstract class can define global attributes for all requirement classes in the same branch. The `Encryption` requirement type, for example, defines the attribute `Algorithm`, which should be used to note the name or identification of an implemented or required encryption algorithm. As all subclasses need this parameter, the attribute can be defined already on their common superclass.

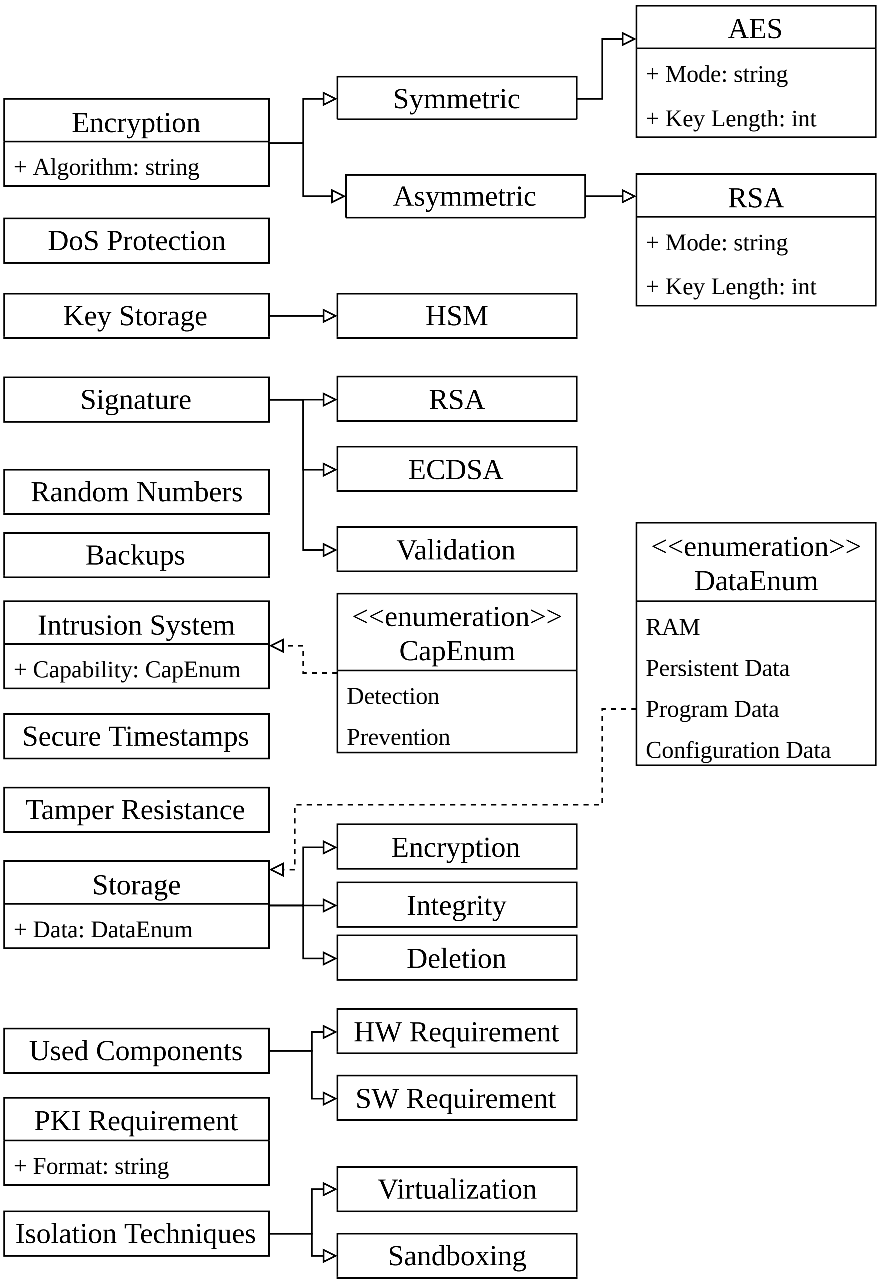


Figure 5.2: Technical Security Requirements Hierarchy [20]

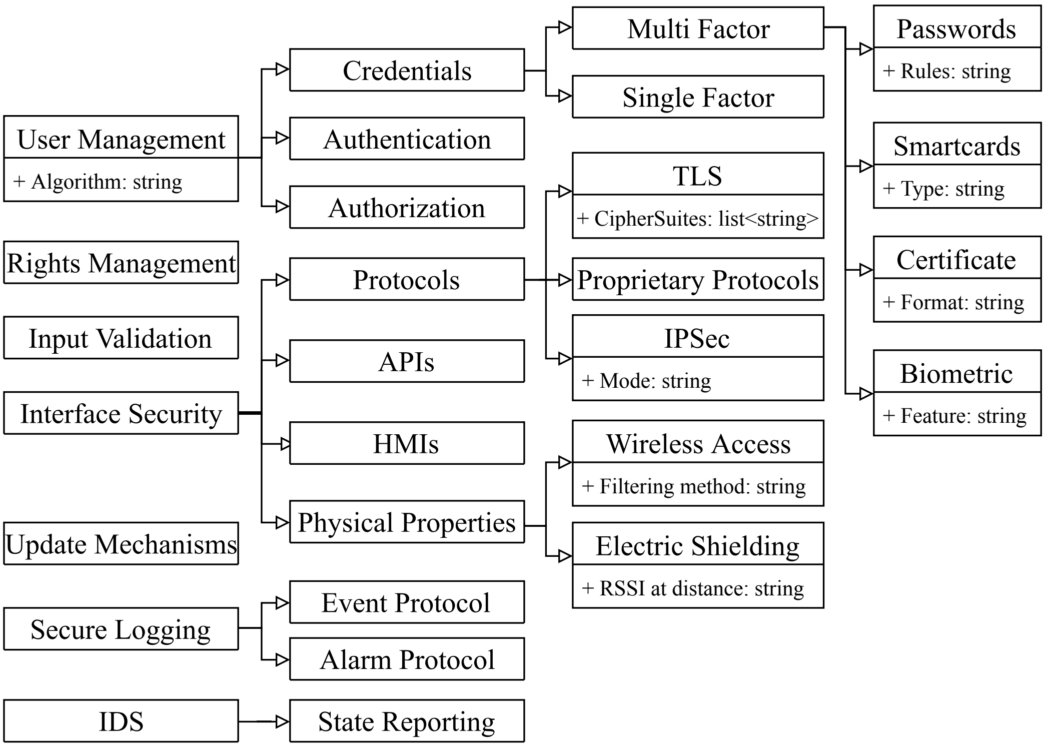


Figure 5.3: Functional Security Requirements Hierarchy [20]

Functional Security Requirements define the behavior of connections to and within the system, including all applied protocols, Application Programming Interfaces (API), and user interfaces. They thereby specify the behavior between outputs and inputs. Figure 5.3 shows a hierarchy of requirement classes within this category.

Process Requirements describe the development process and product lifecycle. They can be initially implemented but hardly retrofitted. A hierarchy of requirement classes within this category is shown in Figure 5.4.

5.2.3 Attributes

The presented data model enables the assignment of global security requirements to the three requirement classes described above. These attributes refer to the base class of a requirement and indicate whether a specific requirement must be certified according to a given standard and how these requirements should be prioritized. For example, as the security objectives in automation industries vary between different scenarios, it is

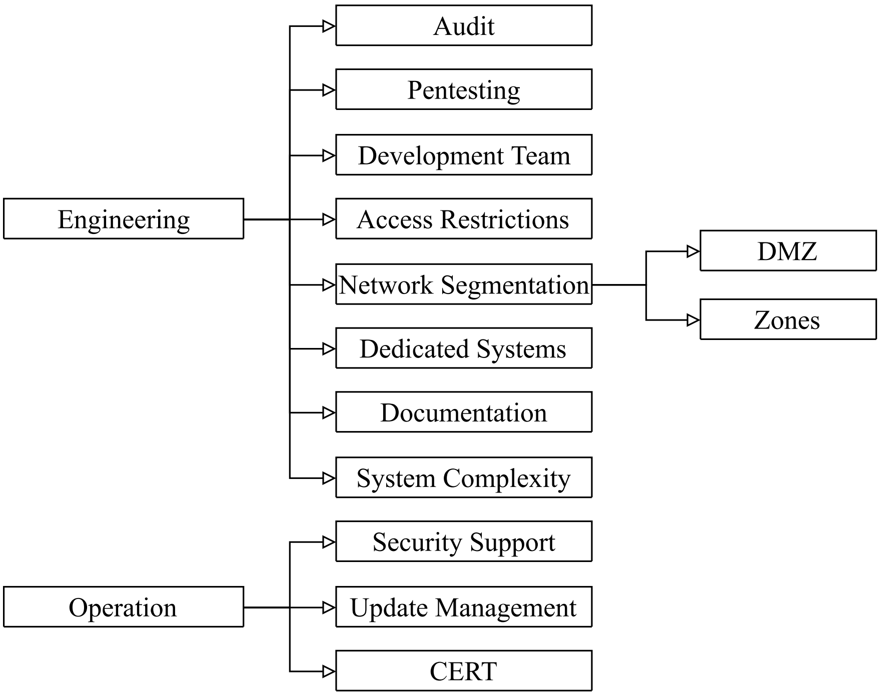


Figure 5.4: Process Security Requirements Hierarchy [20]

possible that the importance of requirements does so, too. An attribute `Priority` can be used to state the priority in which requirements should be met in case of contradicting or not fulfill-able requirements. Similarly, an attribute `CertificationRequired` can indicate whether the fulfillment of this requirement needs to be certified according to the given standard. In addition to these two attributes, further definitions of more specific attributes depending on the domain and field of application are recommended.

5.2.4 Evaluation of the Security Requirements Data Model

A substantial part of the German National Reference Project for IT Security in Industry 4.0 (IUNO) was the development of prototypes for IIoT-specific implementations. The prototypes differ significantly in their applications ranging from customer-tailored production (**PT1**), technology marketplaces (**PT2**), remote administration of production sites (**PT3**), to visual security control centers (**PT4**). During the evaluation of the following IIoT use cases on these prototypes, a total of 416 different security requirements were identified and mapped within the unified security requirements data model [20].

Propagation of Security Requirements in Distributed Supply Chains It is common practice in the manufacturing industry that customers require their suppliers to comply with certain security requirements. Passing on such requirements to the various sub-suppliers along a distributed supply chain is complicated, prone to errors, and causes delays. Each link, respectively sub-supplier, must analyze, implement (where applicable), and forward the received and consolidated requirements from the different customers as a new bundle to their own suppliers. Thereby, each iteration may represent a significant delta to the existing representations that again require manual validation, reconciliation, and merging.

Production Compliance with Given Sets of Process Requirements Security-critical parts of an integrated product, such as the security module of an ECU, must only be manufactured by machines and production sites that meet a bundle of specific security requirements. In the setting of fully interconnected production infrastructures, such a compliance check must be processed in a fully automated way. Only a machine-readable requirement specification allows a production site to autonomously check if it complies with the given set of process-specific security requirements. Therefore, each machine of the manufacturer must present a set of process requirements that it complies with, and further a set of requirements that are fulfilled by the corresponding suppliers, serving the different machines at possibly different production sites. By comparison of the requested security requirements and the implemented capabilities, a managing system must be able to decide whether the manufacturing of the considered product is possible at all.

Automated Product Quality Verification A product that claims to fulfill a particular set of requirements should be validated in a fully automated manner. Such a process enables automated requirements testing as a dedicated service of machines without human interaction. Such an automated requirements verification is further required for standardized verification procedures of given sets of technical and functional security requirements. Therefore, a production line must support automated quality testing of produced parts, as the implemented capabilities are directly compared to the initial requirement sheets like it is common in the software engineering scenarios described in [106, 108]. The number and distribution of identified and mapped security requirements per second-level abstraction class and prototype is shown in Table 5.2.

Table 5.2: Distribution of Security Requirements in the IUNO Prototypes (PT) [20]

Category	**PT1**	**PT2**	**PT3**	**PT4**	**total**
Technical Requirements	71	26	23	56	176
Crypto	11	8	3	5	27
Key Storage	4	3	2	1	10
PKI	1	0	0	1	2
IDS / IPS	1	1	1	6	9
Availability	11	6	8	6	31
Hardening	0	0	0	28	28
Storage	39	5	7	7	58
Secure Time Base	2	2	1	1	6
Random Number Generation	2	1	1	1	5
Tamper Resistance	0	0	0	0	0
Functional Requirements	61	16	36	21	134
Identity Management	30	5	19	12	66
Input Validation	1	0	4	1	6
Interface Security	26	10	8	7	51
Secure Logging	4	1	5	1	11
Update Mechanisms	0	0	0	0	0
Process Requirements	27	39	14	26	106
Engineering	8	21	6	6	41
Operation	19	18	8	20	65
Total Requirements	159	81	73	103	416

Finally, the evaluation showed that the security requirements model is suited to incorporate all so far identified security requirements of state-of-the-art prototypes from the industrial domain, eliminating the need for human interaction.

5.3 Derivation of Intrinsic Security Requirements

Based on the assessed risks, identified threats, and the organization-specific risk appetite, appropriate intrinsic security requirements are synthesized using patterns, formulated in the unified security requirements data model, and prioritized based on the associated security risks.

While the derivation of high-level requirements such as "Confidentiality of the Component Key-storage must be ensured" is straight-forward, the selection of specific implementations, considering the overall structure, is a more delicate task. The first step to automate this process is to develop a threat mitigation catalog that assigns a mitigation to each relevant combination of threat and the base system model object classes.

While an experienced security expert balances the impacts of the risk with possible security measures, an automated activation of security controls bears the risk of misjudgments. Therefore, decisions to take active measures that could negatively affect the functionality of the system shall remain the responsibility of a human controller. However, actions without such risks and preparations to support the decision of the controller should be implemented immediately. The shown approach proposes such measures based on a model of the SuE and pattern for recommendations to support the operator in making proper decisions. In contrast to static networks where it was sufficient to assess the infrastructure once and then inherit security goals from the data, this process should be triggered on each update of the model. Especially for dynamic structures, attack, and impact vectors from which a specific risk was derived should be considered when choosing and prioritizing an appropriate security risk mitigation.

5.3.1 Implementations per Security Objective

The type of appropriate implementation depends on the relevant threat and object classes. In addition to the following description of possible measures and their interaction, Table 5.3 provides exemplary security controls for each of the four basic system object model classes. Furthermore, sets of hierarchical ordered threat mitigations are provided in the Figures 5.2, 5.3 and 5.4.

Table 5.3: Examples of Security Controls and Measures against a targeted Asset

Threat	Function	Component	Connection	Data
Spoofing	Cryptographic Signatures		Segmentation	Cryptographic Signatures
Tampering	Encapsulation, Obfuscation, Monitoring	ACL, Tamper Protection	Segmentation, Authentication	Cryptographic Signatures
Repudiation	Logging	Digital Identity	Logging	Logging
Information Disclosure	Obfuscation, ACL	Storage Encryption	Encryption	Encryption, ACL
Denial of Service	Watchdog, Error Management	Redundancy, Recovery	Redundancy	Redundancy, Fall Back Values
Elevation of Privilege	Hardening, Testing	ACL, Sandboxing		

Supplementing security controls with specific information enables the automated generation of rules for network- or host-based monitoring that may be applied immediately by an Intrusion Prevention System (IPS). However, while ACLs are recommendable in most cases, further firewall rules and active intrusion handling measures should only be automatically prepared and suggested to the operator.

Confidentiality: Confidentiality is threatened by information disclosure. For data, this is avoided by encryption on the producing functions, the transferring connections, and the storing components. Functions containing valuable know-how also have inherent protection needs that can be maintained by obfuscation, by controlling access to binaries and preventing runtime analysis, e.g., via encapsulation. Similarly, protection of the operating parameters and details about components can be achieved by access control or sandboxing. Information about routes, technologies, and performance of connections can be obfuscated, e.g., by making them transparent to high layer applications, or by hiding them in side-channels.

Integrity: The integrity of connections can be ensured by network monitoring [144]. Validation can ensure the integrity of data [127]. For functions, performing reviews and software tests can be a viable measure to initially ensure their integrity. At runtime, their correct execution can be protected by isolation or hardening, which is also a measure to protect the integrity of components.

Availability: The availability of data can be ensured by fallback values, or redundancy in terms of a secondary connection. Therefore, information about the availability of a connection can also be gained by link detection. For functions and components watchdogs, and timeouts are technical controls, while Service Level Agreements (SLA) are administrative measures, especially for functions provided by third-parties.

Authenticity: The authenticity of data is typically ensured by signatures or by appropriate asymmetric encryption. To validate the authenticity of connections, certificate-based authentication methods, as well as Virtual Private Networks (VPN) are widely used. The authenticity of specific functions might be initially validated by functional tests and by appropriate fingerprints during startup and runtime. Similarly, components can use HSMs to validate their authenticity.

Non-repudiation: Non-repudiation of data is typically ensured by asymmetric cryptography, especially cryptographic signatures, and by tamper-secure protocols like a blockchain.

5.3.2 Selection and Prioritization of Security Controls

Cataloging and rating of security controls enables their automated assignment to security goals that exceed a defined security risk threshold. In addition to the applicability as per se, their effects on the RAP and CILs as well as the dependencies introduced by the security control must be considered. Common dependencies are for example a secure keys management, secure implementations, and a secure network configuration.

Since the automated intrinsic security requirements deduction process considers the protection of each security goal individually, it fully supports the defense-in-depth principle. In practice, this can be appropriate for critical assets, but should, usually be adjusted to the delta of the risk level between protected and unprotected implementations. Practically speaking, adding content encryption for a data that is solely transmitted by an end-to-end encrypted TLS connection between secure components rarely provides a substantial increase of security. While the additional content data encryption complicates the execution of confidentiality threats, the resulting marginal delta in the RAP hardly justifies the resulting costs for more powerful components. Furthermore, each additional function increases complexity and may in turn allow for attacks.

5.4 Management of Extrinsic Security Requirements

Extrinsic security requirements originate from external entities. Their automated management includes checking whether they are already fulfilled or whether they need to be implemented in addition to the existing capabilities.

New processes and regulations can entail additional requirements that must be met by the systems involved. In contrast to intrinsic security requirements, which ensure the development and continuous operation of secure systems, this section focuses on extrinsic requirements from external sources.

Identifying combinations of systems that are capable of satisfying complex requirements, like a production line that provides full data encryption along all ECUs is a challenging, yet hardly solvable task. To this purpose, a unified architecture for security requirements, capabilities, and their matchmaking is required. Such a unified architecture for security requirements enables several new use cases.

During the implementation phase, mandatory extrinsic security requirements should be considered before intrinsic requirements are deduced and implemented, as they may mitigate threats that need no longer to be addressed by additional measures. During operation, they either can be fulfilled by already implemented capabilities or must be added as additional security requirements to the design of according CPS.

Ideally, extrinsic security requirements are received in a machine-readable form. If this is not the case, they must initially be formalized.

A common distribution problem in producing domains is to find production lines capable of specific combinations of security requirements that must be met in order to produce or handle specific sensitive information and components. To address and automate this complex task, an integration into middleware protocols like OPC-UA is required and demonstrated in Section 5.4.2.

5.4.1 Fulfillment Verification

Numerous use cases, such as those for the IIoT outlined in Section 5.2.4 on page 72 require the ability to evaluate whether a given requirement is implemented by a single or combination of multiple CPSs, e.g., by a production line. These process-requirements may relate to the targeted product itself but can also be constituted by the requirements of involved supplied parts. In the setting of fully interconnected production infrastructures such a compliance check must be processed in a fully automated way. Only a machine-readable requirement specification allows a production site to autonomously check if it complies with the given set of process requirements. The process requirements may relate to the targeted product itself but can also be constituted by the requirements of involved supplied parts. Each machine of the manufacturer has a set of process capabilities that it complies with and further a set of capabilities that are fulfilled from the corresponding suppliers serving the different machines at possibly different production sites, as depicted in Figure 5.5 on page 82. Such a verification mechanism can be implemented based on the proposed security requirements data model, using off-the-shelf tools as described in detail in [20]. An exemplary system architecture initially designed for the IIoT-specific OPC-UA is shown in Figure 5.6 on page 83. The server holds the security-requirements data model and a special `Implementations` object holding instances of every implemented capability on the corresponding system or machine. The client does not require any additional data. By browsing the object data type definitions of the server with off-the-shelf clients, the whole security requirement space can be explored. In the next step, the operator generates a list of all needed requirements. As the implemented capabilities of the server are published under the `Implementations` object, the client can retrieve all implementation details. By comparison of the demanded requirements and the implemented capabilities, a managing system can decide whether the manufacturing of the considered product is possible at all, as demonstrated by Algorithm 5.1.

For each initial requirement r out of R, every implementation detail c of C is tested whether c is an implementation of a (sub-)class of r. This allows for both specifications - requirements and implementations - being on different abstraction levels. A requirement is assumed to be fulfilled if one of its possible (sub-)implementations is available. For example, while the requirement could be `Symmetric Encryption` (cf. Figure 5.2), the implementation could be as detailed as symmetric encryption with AES-256 and using CBC-mode. By leveraging the hierarchical structure of the object-oriented data model, the check is possible with standard methods (like `instanceof`) available in most modern programming languages. Additionally, the production planning system (PPS) is now able to choose the best site and machine to use to produce the specific product by comparing the requirement sheet with the fulfilled requirements and considering the global `Priority` attribute. The execution of the described algorithm then answers the question whether the specific requirements can be met.

Algorithm 5.1: Security Requirement Verification

```
Input: List of Requirements $R$
Input: List of implemented Capabilities $C$
Output: True if all requirements are met, False otherwise
Let $x$ instanceof $y$ return True if and only if $x$ is an implementation of a
  (sub-)class of $y$;
foreach requirement $r$ in $R$ do
    found = False;
    foreach candidate $c$ in $C$ do
        found = found $\lor$ ($c$ instanceof $r$);
    end
    if $\neg$ found then
        return False;
    end
end
return True;
```

5.4.2 Integration with Middleware Protocols

Mapping the unified security requirements data model to middleware protocols provides full integration into existing communication stacks from ICT mainframes down to the chip level of embedded controllers. A state-of-the-art middleware protocol for M2M communication is the OPC-UA, which is expected to play a major role in future IIoT production scenarios. Providing the data model in this language enables the communication of security requirements horizontally between distributed production sites and vertically through all layers of production.

To this purpose, the security requirements data model outlined in Section 5.2 is implemented using the OPC-UA language. This implementation enables OPC-UA clients and servers to compare requirements and their implementations as described in Section 5.4.1. OPC-UA provides an abstract base class type called `BaseObjectType`. A base class `RequirementType` be defined as the central starting point for the trees depicted in Figures 5.2 to 5.4 by subclassing the general `BaseObjectType`. From there, all child nodes of the security requirements data model are placed in a subclass of their direct parent node.

Listing 5.1: Exemplary XML-representation in OPC-UA for AESEncryptionType [20]

```
<UAObjectType NodeId="ns=1;i=1005" BrowseName="AESEncryptionType">
   <DisplayName>AESEncryptionType</DisplayName>
   <References>
      <Reference ReferenceType="i=45" IsForward="false">ns=1;i=1004</Reference>
      <Reference ReferenceType="i=47">ns=1;i=6005</Reference>
      <Reference ReferenceType="i=47">ns=1;i=6006</Reference>
   </References>
</UAObjectType>
<UAVariable BrowseName="Mode" DataType="i=12" NodeId="ns=1;i=6005" ParentNodeId="ns=1;i=1005">
   <DisplayName>Mode</DisplayName>
   <Description>The operation mode the AES is using.</Description>
   <References>
      <Reference ReferenceType="i=47" IsForward="false">ns=1;i=1005</Reference>
      <Reference ReferenceType="i=40">i=63</Reference>
   </References>
</UAVariable>
<UAVariable BrowseName="Key␣Length" DataType="i=5" NodeId="ns=1;i=6006" ParentNodeId="ns=1;i=1005">
   <DisplayName>Key Length</DisplayName>
   <Description>The (maximal) key length the requirement supports.</Description>
   <References>
      <Reference ReferenceType="i=47" IsForward="false">ns=1;i=1005</Reference>
      <Reference ReferenceType="i=40">i=63</Reference>
   </References>
</UAVariable>
```

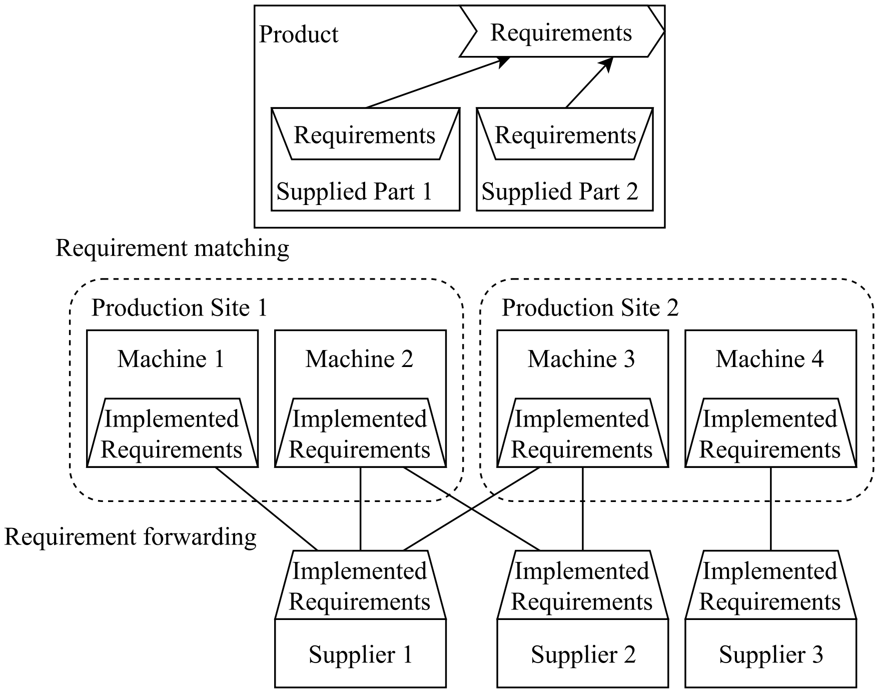


Figure 5.5: Security-related Dependencies in Production Lines [20]

An example of a leaf definition is shown in Listing 5.1. Lines 1–8 define the new object type while the Lines 5–6 refer to the attributes of this object type definition using the OPC-UA reference type *HasComponent* `(i=47)`. Lines 9–16 then describe one attribute in detail. Each attribute can have a different data type, which is referenced by the `DataType` attribute in Line 9. The data type `i=12` of the `Mode` attribute corresponds to the string base type. An addition to those leaf attributes, the `AESEncryptionType` further inherits all attributes of its ancestor classes and as such has quite a detailed description. Being specified in the OPC-UA XML-language, the data model can easily be extended and edited with off-the-shelf OPC-UA data model editors like the OPC-UA Modeler provided by [145].

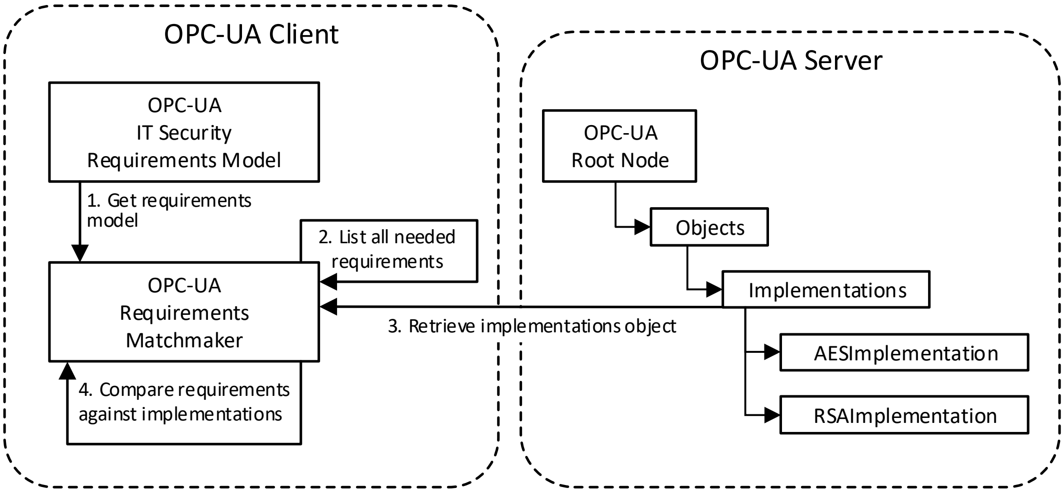


Figure 5.6: System Architecture for Requirement Verification using OPC-UA [20]

5.5 Discussion

This chapter introduced methods for deriving and managing intrinsic and extrinsic security requirements. The hierarchical, object-oriented security requirements data model, introduced at the beginning of this chapter, enables the automated derivation and management of such security requirements.

Documenting, propagating, and tracking requirements for software and systems are common practice now; however, reconciling with requirements from external sources involves regularly considerable manual effort. In particular, manufacturers and suppliers across the supply chain require a unified data model to document and communicate security requirements.

The automated, pattern-based derivation of security requirements supports the creation of uniform requirements and measures catalogs. This method ensures that no risks are missed, avoids the proliferation of non-consolidated ad-hoc solutions, and helps to focus efforts on developing effective remedies. For many cases, several different security controls are proposed, from which the optimal cannot automatically be selected yet, but still rely on expert knowledge. This is owed to the fact that the provided SSML is not sufficiently expressive to describe specific situations. However, this approach seems preferable since the alternative would be an inapplicable and complex ontology. The included exemplary catalog of security measures is meant as a starting point that must be adopted to the needs, capabilities, and use cases of each organization.

In contrast to most other requirements purposes, security requirements to be implemented or already implemented can and should be systematically cataloged in order to reach and maintain a consistent level of protection. Such a consistent level of protection leads to a corporate security culture and can be achieved by a pattern-based inferring from security goals at risk to high-level requirements. This is possible during the system development, following the security-by-design principle, but also for retrofitting.

A problem yet hardly resolvable by automation are conflicting requirements. A real-world example of this is a Chinese national requirement that all Electronic Vehicles (EV) must continuously report their battery status to a remote data center [146]. As this includes personal data, the method infers that this data must be adequately protected, meaning that a connection must only be established after authentication of the data center. In case of authentication problems, this leads to a conflict between the protection of sensitive data on the one hand and the fulfillment of monitoring obligations on the other, which cannot be solved without legal expertise.

While the presented security measures effectively decrease the risks to the SuE, their monitoring for potential anomalies and handling of incidents, which violate the set requirements and demand an iteration of the SRA are subject of the following chapter.

6 Threat Detection, Correlation, and Response

When incidents are detected, vulnerabilities disclosed, or IoC received, their impact on the overall security situation must be determined, and an appropriate response implemented. Since such events can mean a change in the risk landscape, they require a short-term update of the security risk assessment. Addressing the question on how to detect, analyze and respond to potentially disruptive changes of the security situation (**RQ 4**), this chapter introduces a set of appropriate tools to detect, categorize and correlate incidents, and to reflect the resulting insights into the model for the SRA, introduced in Chapter 4. This leads to a semi-automated pipeline from incident and vulnerability detection to automated risk reevaluation.

For this purpose, concepts from my following earlier publications are adopted and improved.

The concept of CPS-optimized acquisition and forwarding of network captures was published as paper “Packet-wise Compression and Forwarding of Industrial Network Captures” [22], which I presented at the *9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* in 2017.

The concept of combining machine learning with a hierarchic ontology for incident classification, forwarding, and correlation to form a pan-European early warning system for large-scale cross-company attack campaigns was published as paper “Detecting and Correlating Supranational Threats for Critical Infrastructures” [27], which I presented at the *15th European Conference on Cyber Warfare and Security* in 2016.

6.1 Introduction

“No plan survives contact with the enemy.”, Helmuth von Moltke the Elder

Since CPSs within critical infrastructures have become strategic targets for advanced cyber-attacks, the provision of new defense techniques for their protection is a severe challenge. Despite all taken measures, there is always a residual risk of successful compromise, resulting in a critical situation. Interconnection across multiple networks make CPSs vulnerable to network threats like an intrusion, exploitation, data extrusion, or malware. Thereby, the main challenges are to detect related incidents, analyze them, and draw reasonable conclusions.

The detection of anomalies in network traffic requires making network traces available to the monitoring system. The analysis of detected incidents is necessary, primarily to determine their full extent, and secondary to identify IoCs that may be present in other affected systems but have not yet been recognized there. In a meshed ecosystem, it is further essential to identify and prevent further compromises, within but also outside one’s own organization. The detection of large-scale campaigns is, in particular, essential for critical infrastructures and distributed CPSs.

Unfortunately, the absence of detected incidents does not imply that there is none. One way to improve overall security is to exchange IoCs to look for, such as (small) time deviations in the network traffic that may point to a third-party device altering values in a vehicle, or hard to detect but easy to spot deviations in the performance of a specific CPS like a centrifuge [147]. Problems in the exchange of such IoCs are the fear of organizations to suffer disadvantages by publishing incidents and disclosing valuable information about their technologies to competitors. Therefore, the unlinkability and limited release of incident information on the need-to-know principle are essential for organizations to provide detailed incident information. In particular, a limited release of incident information is required to avoid market distortions but enable large-scale correlation on the extend of campaigns.

A common practice to detect cybersecurity incidents in enterprise environments are SIEM and NIDSs that monitor and validate transmitted packets but hardly meet the specific requirements of CPS environments. To generally improve this situation, a distributed supranational architecture supporting the automation of detection, classification, and mitigation of highly sophisticated cyber-attacks, targeted simultaneously at multiple infrastructures, is presented. Upgrading IDS and SIEM solutions to better support CPS enables orchestrated incident management and correlation by a three-layered architecture comprised of SOCs at organizational, national, and European level. The approach combines machine learning and semantic reasoning: First, methods from the field of machine learning are applied to analyze threat indicators of different granularity, providing a classification of very specific observables collected at compromised sites. Second, an analysis is performed to identify large scale correlations within an incident knowledge graph, yielding insight into ongoing attack campaigns, especially regarding the extent and expected impact. While optimized handling of network traces improves local anomaly detection, the distributed incident communication architecture counters advanced threats targeted against critical infrastructures by allowing the identification of potential targets, which are likely to be affected or already compromised.

6.2 Packet-wise Compression and Forwarding of Network Captures

While there are plenty of generic and domain specific NIDS, providing them with the relevant data for proper analysis is a further challenge. Techniques for extracting and providing this data are discussed in the following Section 6.2.1. The required bandwidth and the amount of incoming data make transmitting all network records from CPSs in the field, e.g., for a fleet of more than a million vehicles, to a remote NIDS an extraordinary challenge. The thereby introduced side-channel further facilitates eavesdropping attacks since the attacker can choose the more convenient channel for tapping, direct between the functional sender and receiver, or on the connection to the NIDS. An approach to significantly reduce the amount of exchanged data and provide some level of protection is presented in Section 6.2.2.

6.2.1 Network Traffic Capturing, Forwarding, and Storing

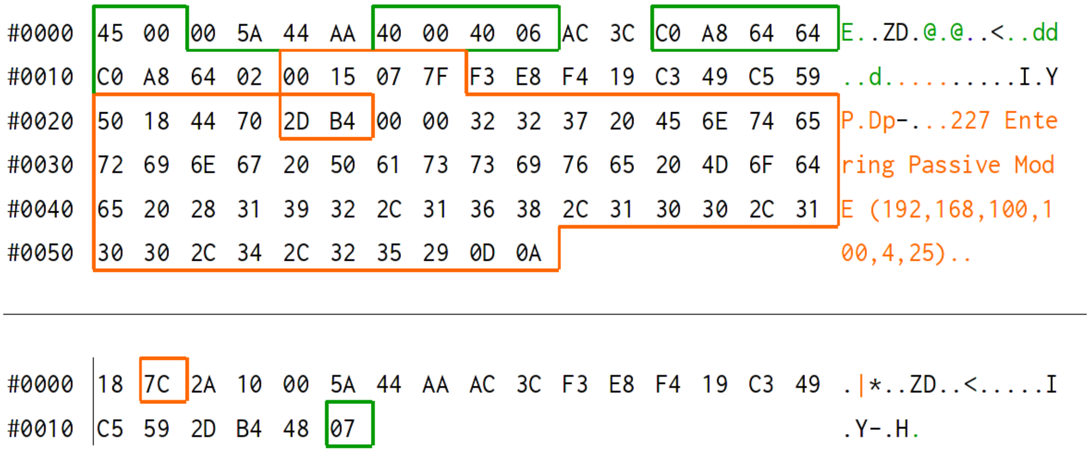
Network capturing is the first step in acquiring useful datasets for further processing. To maintain sufficient quality, it must be ensured that all networks of the monitored nodes are recorded. To protect integrity, the monitoring system should be passive, i.e., it should have a write restriction to the line, so that the operation of monitored systems could not be impaired at any time. Depending on the media and the connected components, additional hardware interfaces might be required for the capturing. Ethernet oriented communication systems can be tapped using standard network capturing software like Wireshark, tcpdump, or scapy. In the case of serial network protocols, a conversion to a packet-based format is required for IP-based forwarding. On the hardware side, network probing, port mirroring, or software modules can be used at the communication endpoints to probe the network. A Network Wiretap Device (TAP) is a probing device that enables recording of the data transmitted across a network cable. Placed in front of a monitored device, it can capture all sent or received packets but should not be able to actively interfere with the monitored communication. Generally, TAPs are a viable choice to integrate a monitor feature into legacy devices and infrastructure, as no modification of existing components is required. Port mirroring is another technique, which can be applied either at the network switch (in a star-topology) that connects the monitored device or at the port of the device itself. While wiretapping is often easier to implement, mirroring the receiving port guarantees that all packets are recognized precisely as the machine receives them. Due to the required modifications, port mirroring is viable for modern infrastructures or new devices but rarely an option for legacy environments. A technique similar to port mirroring is to tap the data by a software implementation. In this case, a specialized library aware of every network API call could be used to collect the communication packets and send them to an aggregator. This option is viable for the outgoing traffic of a control system where only specific packets have to be captured. To allow forwarding of all incoming data, the monitoring network should be faster than the monitored one, as the captured data is enriched by additional meta-information like timestamps and system-conditions. As the captures are by now in a packet-based state, they can be forwarded via standard TCP/IP communication, using state-of-the-art encryption libraries where needed. The deduplication described in the following section needs to be done before any encryption. While it can be necessary to restore captures to their original form for further analysis, it is also a memory friendly option to store them in a (time series) database in the compressed form, together with the decompression scheme.

6.2.2 Data Minimization by Deduplication

One approach to reduce the size of the captured data and minimize the transmission of potentially confidential information is to replace frequent and critical content with IDs from a lookup-table. This causes the network traffic to decrease over time because all frequent network packets should become known in the lookup-table. When a new packet is detected at the sensor, variable sections are determined and transmitted unchanged, while the rest of the packet is merged into one sequence.

6.2.3 Private Encoding

A positive side effect of the deduplication method is the concealment of the retransmitted data. More important, an eavesdropping attacker could not identify the original sender and recipient of the packets, as the IP header deduplication covers this information and omits them from the transmission in over 99% of the packets. An example for this method is shown by Figure 6.1. The upper part of the figure shows the original packet, while the deduplicated version, which may be eavesdropped, is shown below.



```
#0000 45 00 00 5A 44 AA 40 00 40 06 AC 3C C0 A8 64 64 E..ZD.@.@..<..dd
#0010 C0 A8 64 02 00 15 07 7F F3 E8 F4 19 C3 49 C5 59 ..d..........I.Y
#0020 50 18 44 70 2D B4 00 00 32 32 37 20 45 6E 74 65 P.Dp-...227 Ente
#0030 72 69 6E 67 20 50 61 73 73 69 76 65 20 4D 6F 64 ring Passive Mod
#0040 65 20 28 31 39 32 2C 31 36 38 2C 31 30 30 2C 31 E (192,168,100,1
#0050 30 30 2C 34 2C 32 35 29 0D 0A                   00,4,25)..

#0000 18 7C 2A 10 00 5A 44 AA AC 3C F3 E8 F4 19 C3 49 .|*..ZD..<.....I
#0010 C5 59 2D B4 48 07                               .Y-.H.
```

Figure 6.1: Example of the Deduplication Effect on an FTP Packet [22]

The marked parts resemble the original data and their substitution by an ID. Most valuable information, i.e., sender and recipient addresses, ports, and actual payload, cannot be derived directly from the deduplicated packet. Using this online deduplication approach, an attacker is not directly able to infer the meaning of the transmitted IDs. As the IDs are only once transmitted with their corresponding byte sequence, an attacker is only able to reassemble the original packets if he already captured this packet during the learning phase. In practice, this leads to a system that is vulnerable during start-up and becomes highly specialized to the needs of the sender and receiver over time while building up a customized protocol of IDs, which are private only to the sending and receiving node.

6.2.4 Discussion on the Packet-forwarding Approach

Since the majority of industrial network captures consist of frequent, low-entropy status messages, connection specification deduplication enables significant traffic reduction while providing additional privacy by preventing content information from repeated transmission. As shown in Table 6.1 of [22], deduplication in combination with zlib or alone performed better than the traditional, stateless zlib compression.

Table 6.1: Data Reduction Evaluation for Different Datasets [22]

Dataset	Number of packets	Original size	zlib ratio	Deduplication ratio	Combined ratio
Modbus [125]					
run8.pcap	72186	5008499	0.96	0.88	**0.86**
channel_2d_3s.pcap	383312	17449820	1.17	**0.71**	0.90
run11.pcap	72498	4955264	0.96	**0.87**	0.87
run1_3rtu_2s.pcap	305932	15870003	1.05	**0.69**	0.81
4SICS - S7 [126]					
151020.pcap	246137	18000708	0.91	**0.36**	0.46
151021.pcap	1253100	101191303	0.85	0.55	**0.53**
151022.pcap	2274747	139975880	1.02	**0.71**	0.78
Office Network [124]					
KDD’99 Day 1	1362869	280299459	0.81	0.93	**0.76**
Self Recorded Robot					
lab.pcap	10120	3227398	0.39	0.95	**0.36**

The reason for this is that industrial network protocols, unlike office network protocols, use small but very frequent packets. Thereby, the limited set of header field combinations, like source, destination and parameters, but also frequent payload are efficiently substitutable parts. For high entropy data like encrypted network traffic, or the fast-changing position reports of a production robot, deduplication of packet payload is hardly effective.

Even in such cases, the reliable savings through deduplication of packet headers prevents an overhead in the long term.

While the usage of offline compression algorithms can achieve a much higher reduction of data size, these require knowledge of all data to be processed, i.e. they rely on processing the data in blocks. Doing an online data deduplication only based on prior packets allows for removing redundancy between packets while future data is still unknown. In cases where adequate protection is required due to confidentiality of the captured data, the concealment substitution of process specific parameter to unspecific identifiers is often not sufficient, as an attacker that monitors the connection from the beginning can also create the corresponding lookup-table. Instead, conventional measures like TLS encryption are necessary and applicable, as they do not interfere each other.

6.3 Incident Categorization and Threat Correlation

The increasing levels of interconnection and interdependency of CPSs have significantly enlarged their attack surface. Disruptions of vital systems such as energy, communication, or transportation may substantially affect modern society. While there has been research effort for protection of isolated domains within single nation states, it is still an open question how to effectively implement a pan-European incident management system handling threat detection, large-scale attack correlation, and early warning. This becomes even more challenging in the light of multistage attacks exploiting interdependencies. Advanced threats such as Emotet [148], Stuxnet [147], and Dragonfly [149] are on the rise while the extent and full impact of future similar attacks are not predictable at present time. Only performing large-scale threat detection on a, e.g., pan-European level will provide full situational awareness regarding cyber incidents. Threat information highly varies in granularity and completeness. Noisy and low-level threat information gathered within single sites provides detailed local information but misses the relationship to the overall threat situation. In contrast, large-scale correlation technologies are not suited to process the vast amount of detailed and noisy threat information data collected locally. To fit local information into the overall context, machine learning classification for low-level data is combined with semantic reasoning, facilitating situational awareness, early warning to possibly affected critical infrastructures and optimal mitigation strategies.

6.3.1 Feature Diversity

In this section, the diversity of fine-grained and coarse-grained features for threat analysis are exemplary highlighted. Potential feature sources are described in [150].

Fine-Grained Features

There is a magnitude of low-level features that would be suited for decision tree classification. While the features for a real-world implementation need to be selected according to evaluation of the running detection system, some exemplary features are presented here for illustration.

Non-Executable File Features Features for the classification of non-executable files may include meta-data and detailed characteristics of the file. The choice of features strongly depends on the specific deployment of the classification module. For example, if the malicious file is an instruction list for a field level PLC, features could be identifier, input, mode, time basis, programmed value, actual value, and modifiable flag of specific sections.

Executable File Features Features for executable files may include characteristic features such as Application Binary Interface (ABI), system calls, type of subroutines (File I/O, Threat, Network, GUI, Registry), number of branches in specific sections, exported functions, or mutexes.

Network Traffic Samples Features of network captures may include information about packet sizes, throughput, session frequency, flow direction (depending on the initiating system of the communication), protocol and protocol settings, and entropy of messages.

Sensor/Actor Features Typical features for sensor and actor communication are control message frequency and set values (e.g. temperature values in a predefined time, liquid level, or voltage).

Coarse-Grained Features

Similar to selecting low-level features, there is a multitude of choices for high-level features that can be selected for reasoning. The following features are exemplarily chosen, motivated by [119] and shall give an impression about the various possibilities.

Attack Vector and Line of Action Based on the incident analysis, it is important to reconstruct the initial attack vector as part of the overall line of action the attacker used. While most incident reports provide only the initial attack vector of an attack, the exact line of action supports the attacker recognition as they often follow an individual procedure (same moving pattern, tools, and methods) to perform their tasks across different victims.

Infection and Stealth Mechanisms Infection and stealth mechanisms such as Master Boot Record (MBR) infection, hiding in partitions, or interrupt and message hooks can also be considered. Further features for large-scale correlation are encrypted network communication, number of distinct IP addresses, steganographic capabilities (e.g. DNS tunneling), and fast-flux exfiltration.

Injection Targets Injection Targets may include specific servers and services, infected databases, targeted sensor and actor components, and network infrastructure (e.g. connection to Enterprise Resource Planning (ERP) systems).

Target Environment The target environment features of an attack may include information about the domain and zone where it is installed and operates, the connectivity and the surrounding ICT infrastructure.

Time and Periodicity Attacks often follow a characteristic timing behavior. Malicious programs for example communicate to their command and control servers in predefined time frames. The periodicity and timing can reveal valuable unique patterns. Another example is given by the timing of multi-stage attacks. If there are several local IoCs in a row, the timing of such a sequence may reveal a particular already known threat pattern.

Impact The impact features include information about the local impact on the affected machine(s) like a loss of functionality, manipulation of processed or displayed values, remote control, malware spreading, or a permanent destruction of the component. The observation of several known IoCs at different devices within one group may also reveal an already known threat pattern.

Interdependencies to Critical Infrastructures This feature includes information about relations of parts or services on which other critical infrastructures depend. This is necessary to correlate attacks and identify the direct and indirect affected victims. It is especially important, as it is possible that the attack was just a single step of a large-scale campaign with the aim to disturb the operation of another critical infrastructure by breaking its supply chain.

6.3.2 Feature Harmonization

To get overall situational awareness data from low-level sources must be fused with abstract and general environment data. For example, if a malicious executable is found by a locally installed SIEM system and this local information is to be fitted in the pan-European context, fine-grained information about the malicious executable must be combined with high-level information data: Where has this file has been sighted in the past? Under which facility environment? Which mitigation strategies where applied in reaction? In a twofold approach to the analysis of threat information, low-level data is first processed and classified locally using methods from the field of machine learning. Second, semantic thinking is used to understand how to integrate the results of this classification into the overall threat landscape. Classifying a given sample means assigning predefined labels. If there are multiple labels to be assigned to the sample, the approach is referred to as multi-label classification, while for more than two possible classes that can be assigned to the sample, multi-class classification methods are used. In a supervised learning setting, the classifier is given a set of training samples together with corresponding labels. In the learning step, the classifier processes this training set to adapt its parameters. When given an unknown sample not in the training sample set, the trained classifier assigns a label to the sample based on the preceding training step. This way, unknown and noisy samples can be classified. There is a diversity of classification methods that suit the purposes of handling noisy fine-grained features, e.g., linear classifiers, support vector machines, kernel methods, neural networks, decision trees, or random forests. Regardless of which method is applied, the classifier takes as input a set of fine-grained features and a sample threat to be classified and outputs a set of classes. Such a mapping from fine-grained features to class labels provides a common level of abstraction for the IoCs, which is necessary for subsequent semantic reasoning. In general, classification of low-level features of suspicious files or network traffic can detect labels of a predefined set. The detected classes are then redirected to the semantic correlation module (OM).

6.3.3 Threat Correlation

The threat labels gained from fine-grained feature classification are now on a comparable level of abstraction with other IoCs gathered from SIEM and IDS solutions. This enables correlations with previous threats by semantic reasoning. To correlate threats, coarse-grained features as presented in Section 6.3.1 must be aggregated and processed. Knowledge graphs archive and relate the detected class labels, coarse-grained feature vectors, and IoCs. This enables the formulation of a variety of queries to search for connections and patterns.

A BOK hosted and maintained by the Security Operations Center at European Level (E-SOC) serves as a growing repository of technical information about incidents, samples, coarse- and fine-grained features, and the relations between them. All information is stored in a unified manner using a hierarchical incident ontology, extended with IoCs and system feature classification schemes. The BOK serves search requests from each SOC where required. The identity of the information source (the affected operator) and all related references are stored pseudonymized and only get disclosed when a substantial interest is justified. By accessing the BOK, previously identified attack patterns are recognized, correlated and conclusions about the expected attacker behavior and their final target can be drawn. For example, an operator recognizing an attack can send requests to the BOK regarding the expected next steps of the attacker, the impact to be expected, or the potential final aim of the attack. An individual Local Knowledge Graph (LKG) is hosted and maintained by each SOC. It serves as a memory for restricted or sensitive individual information like detailed system descriptions and configurations, business dependencies, and organizational structures. Security Operations Center at National Level (N-SOC) and E-SOC particularly are aware of interdependencies of supervised CPSs and know their respective technical services. While direct relations between two operators are rather obvious, it becomes complex to identify linkages across a whole supply chain. This is because dependencies between operators can appear not only in directed forms like supplier-consumer or service-provider, but also include cases where two or more operators depend on each other. Therefore, it is necessary to track and solve interdependency information across all SOC layers.

By using machine learning technologies to fit information into an incident management ontology, consistent relations and classes that enable an automated analysis can be received. Further, implicit information in the knowledge graph is made explicit by inference reasoning and can then be retrieved using a semantic query language. Automated reasoning results trigger Computer Security Incident Response Team (CSIRT) on incidents that would otherwise stay undetected, like e.g. accumulation of incidents sharing a minor common feature. This solution includes different threat detection and correlation steps to be performed at each SOC level. The necessary information is exchanged using STIX [136] messages in combination with RDF/XML triples in the common ontology.

In order to derive context information, the semantic reasoning module performs an inference from the detected features and the information from the LKG and BOK. During all phases of the analysis, each new information is submitted to the N-SOC, including all IoC, classes, performed measures, and information about the affected systems and services. This processing enables a more efficient local incident handling compared to a sole CSIRT that tries to handle the actual situation.

The correlation module supports the N-SOC with a correlation of incidents on a national level, identification of expected impacts, and issuing of specific warnings to operators of critical infrastructures. To find common patterns between distributed incidents, the features labeled by the classifier and reported by the Security Operations Center at Organizational Level (O-SOC) are matched with those of prior and current situations using the semantic reasoner. By correlating the received incident notifications from multiple O-SOCs, the N-SOC can gain insight regarding the severity and extent of the campaign, further potentially vulnerable systems, and reveal hints about possible attackers. Operators of potentially vulnerable CPSs are immediately alerted by the N-SOC to watch out for the found IoC and to take appropriate actions as described by [94]. Beside the incident management support, the N-SOC also serves as a filter for messages between the E-SOC and the O-SOC. It forwards information about features, classes, and relations detected by the O-SOC to be added to the BOK by the E-SOC. During this process, it generalizes sensitive features such that they reveal no sensitive business information about the O-SOC. In the other direction, the N-SOC forwards warnings from the E-SOC to the O-SOCs where required. Further, the correlation module supports the E-SOC on its task to monitor and coordinate the activities of the N-SOCs. Therefore, it supports the detection of (large-scale) attack campaigns and the issuing of specific warnings to operators of critical infrastructures. Additionally, the correlation module resolves dependencies between different critical infrastructure domains. As the N-SOCs submit only non-sensitive and generalized (the classes of sensitive context) information due to privacy reasons, the E-SOC operates on coarser-grained information. The E-SOC searches large-scale attack indicators by correlating input from the N-SOCs, the LKG, as well as the common BOK. Thereby, accumulations of striking patterns such as domains, areas, and timings, as well as targeted supply chains can be detected. The STIX alert messages sent from the E-SOCs to the N-SOC include a list of classes and IoCs that are assumed to be at risk. This leads to a situation specific early warning for operators of similar or dependent systems across borders without revealing sensitive information about threats or operators.

6.3.4 Architecture and Data Flows

The overall architecture of the correlation module (CM) is depicted in Figure 6.2, Monitoring modules (M) of each system forward the recorded samples to the feature aggregator (FA). Coarse-grained feature values are directly extracted and forwarded to the semantic reasoning (SR) module. Fine-grained features are first classified in the Fine-grained Feature Classification (FFC) module. The resulting class labels are then forwarded to the OR module. The coarse-grained features values as well as the class labels for the reported incident are then correlated to the LKG and the BOK. As depicted in Figure 6.3, each SOC deploys its own CM. The correlation modules share one BOK. In practical implementations, this BOK is mirrored, to provide backup and recovery in order to minimize the risk of a potential outage of this central component. The O-SOC CMs forward both, suspicious and confirmed IoCs to the respective N-SOC, which takes the received data to perform correlation of national threats. The N-SOC then forward correlation results to both, the E-SOC and the related O-SOC. In case the FFC of an O-SOC cannot label a given sample to a predefined class, it forwards the monitored sample to the N-SOC. Similarly, if the N-SOC FFC cannot find a fitting

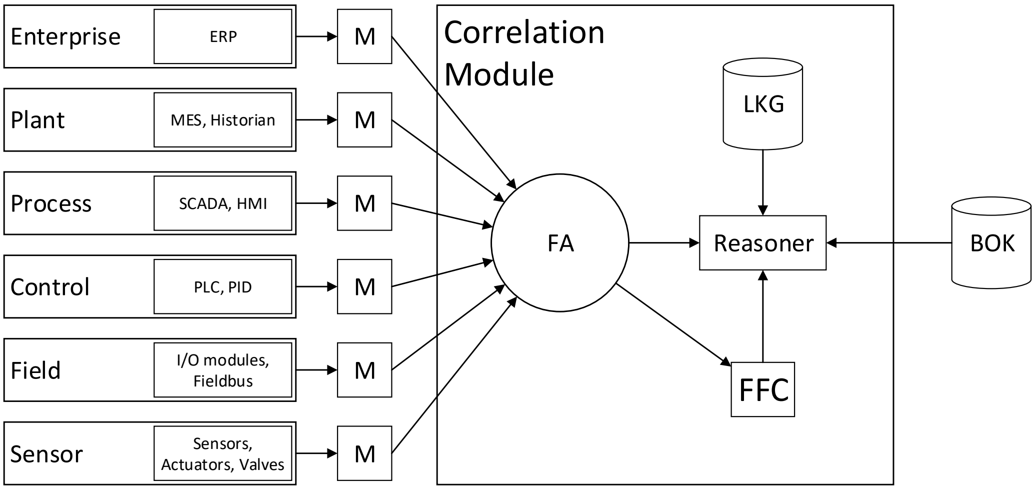


Figure 6.2: Architecture of the Threat Detection Module

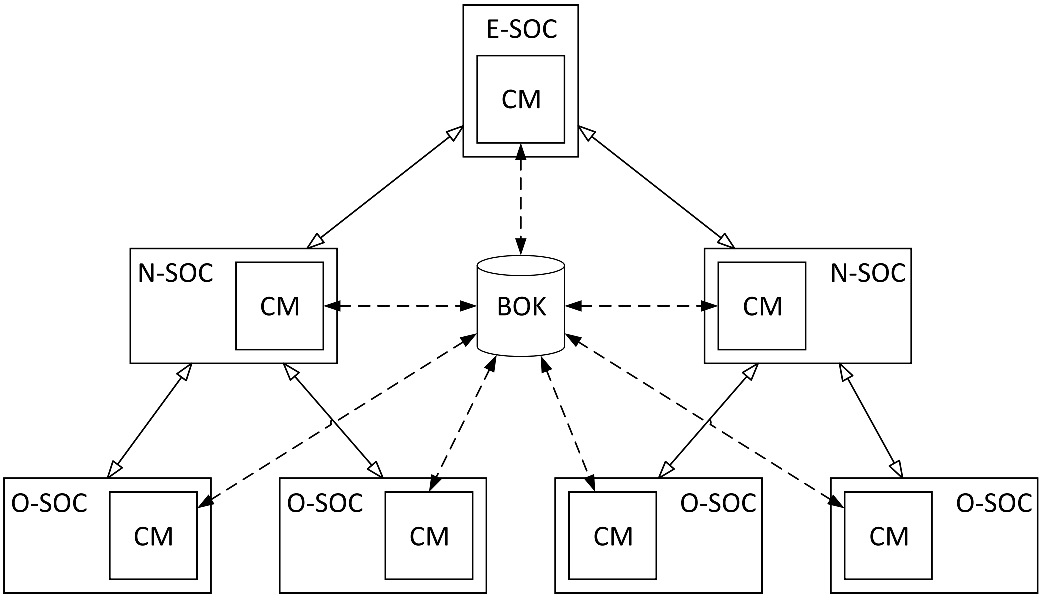


Figure 6.3: Communication and Access Structure

label, it forwards the sample to the E-SOC. This way the N-SOC and the E-SOC are capable of finding and defining new class labels for anomaly patterns that are unknown to the O-SOC. The O-SOC FFC parameters for classification and class label lists are regularly updated with the findings of the N-SOC, and analogue the N-SOC FFC receive updates from the E-SOC.

6.3.5 Discussion on the Incident Correlation Approach

The proposed method includes a definition of common, machine-readable levels of abstraction by transforming fine-grained features into coarse-grained classes. This classification pre-processing allows correlating threat information to potential attack patterns. Correlation is done at operational, national, and European levels, and local and global knowledge graphs are defined in order to provide separation of sensitive information and sharing of correlation insights with potentially affected operators. While classification is provided by methods from the field of machine learning, semantic reasoning is used to correlate and detect new attack patterns. The knowledge graphs store IoCs, which enables the formulation of a diversity of queries in order to search for correlations and patterns. As new correlation insights are directly propagated to the N-SOC and E-SOC, the system provides a basis to cast early warnings to dependent critical infrastructures.

6.4 Reflecting Changes in the Security Landscape

A proper response to security-relevant insights requires a short-term examination and assessment of their influence on the current situation. From a risk perspective, changes in the security landscape may affect either the impact or the likelihood of risks. The first step thereby, is to determine its particular relevance for the SuE. An automated determination requires unified identifiers, such as provided by the Common Platform Enumeration (CPE).

Changes to the impact are typically not bound to a specific implementation, but occur due to updated ratings, new use cases, or processes that impose penalties. A usual reason for changed ratings in the automotive and industrial domains are carry-on-parts, that are used in a platform concept for different products, such as car models. Updating the impact ratings in the model is straightforward by modifying the affected CILs and propagating the changes along the dependency tree.

The likelihood can change due to deviations in the attacker model, the system model, or the threat mitigations. Additional attacker capabilities can cause severe distortions, e.g., by disclosed information, when expensive, bespoke tools can be substituted cheaply, or a vulnerability can be exploited. Depending on the individual situation either the affected threat capabilities, or (more likely) threat factors of the modeled implementations must be adjusted. In the second case, it is necessary to reassess the relevant threats to consider the individual effect on each affected security goal. Given a sufficiently detailed model, this process can be automated by inferring the effect to each affected implementation [23]. Alteration of the overall SuE can create or eliminate potential attack vectors. In such cases, the model must be updated to reflect the new structure and the individual assessments of the changed implementations. The impact propagation is thereby performed like for a new model. While the practical effects of invalidated threat mitigations can be manifold, the required alterations of the model are comparatively simple. In case of a complete ineffectiveness, the *mitigates* relation between the `Threat` the `Threat Mitigation` has to be removed. In the more common case of changes of the capabilities required to implement a threat, the relevant risk factors must be adjusted. A demonstration for such an adjustment is shown in Section 7.4.3 of the following chapter on the evaluation of the presented methods and processes by a realistic use case.

7 Use Case Evaluation

This chapter demonstrates the application of the security risk assessment and requirements derivation methods and procedures introduced in Chapter 4 and 5, by a simplified but recent use case from the automotive domain.

Initially, the context of the use case, a regenerative vehicle brake, is established by an overall introduction to the SuE in Section 7.1. Next, a model of the brake system is created in Section 7.2 and analyzed for security risks in Section 7.3, both as described in Chapter 4. Subsequently, the application of the methods and processes described in Chapter 5 for deriving and managing security requirements, and in Section 6.4 for the mapping of vulnerabilities, is demonstrated using the work product of the SRA in Section 7.4.

7.1 Introduction: Regenerative Vehicle Brake

Regenerative brakes are a homogeneous part of the energy management of electric and hybrid vehicles, including buses.

By converting kinetic energy into storable electricity, regeneration significantly reduces fuel consumption and brake pad wear. In contrast to conventional braking systems where the torque is converted into unwanted and wasted heat due to friction in the brakes, it is used here to reload the batteries. This concept is taken to the extreme by regenerative commercial vehicles such as the Swiss eDumper dump truck [151], which uses the energy collected during a loaded downhill run to drive back up again unloaded. Unfortunately, regenerative brakes in regular vehicles neither work with a fully charged or hot battery nor is their braking torque usually sufficient to ensure the necessary driving stability under traffic conditions. A detailed discussion of this technique is provided by [152]. In addition to deceleration, the braking system can release additional acceleration torque reserved for traction control.

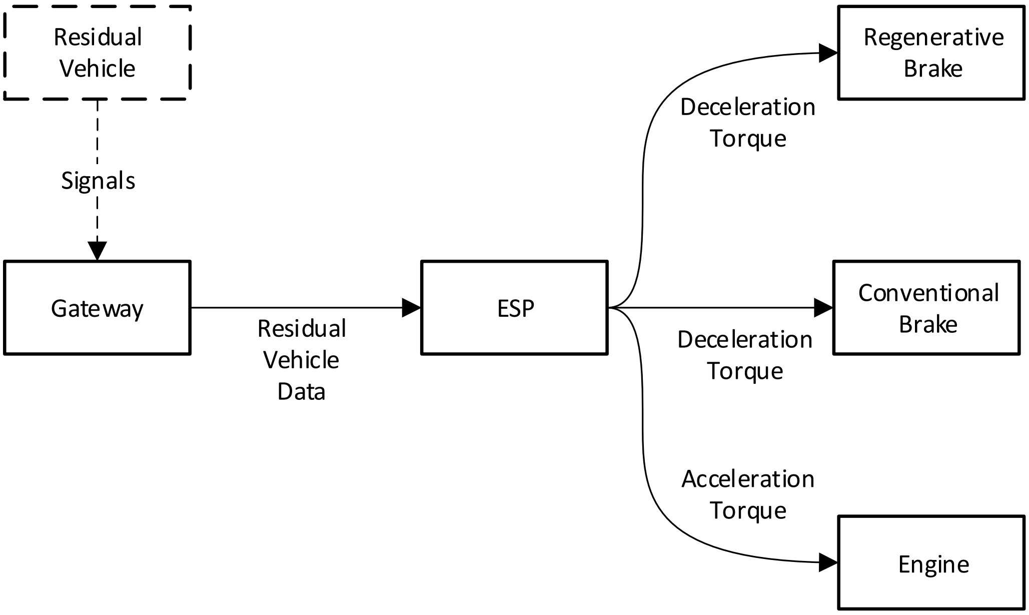


Figure 7.1: Use Case - Regenerative Brake Architecture

From a security perspective, the regenerative brake is a quite interesting example of a sensitive CPS. It attracts technically interested vehicle owners, professional tuners, saboteurs, and competitors alike despite severe consequences. Vehicle owners and tuners want to unlock torque or increase energy recovery, both at the expense of driving stability. Saboteurs target the integrity as a brake can decelerate, but even worse, accelerate as part of the stabilization program. Besides, competitors may be interested in reverse-engineering the braking system to gain valuable know-how about optimal configurations for comfortable and efficient motion management.

Based on the architecture shown in Figure 7.1, the use case sequence of a typical braking situation is illustrated in Figure 7.2. The brake trigger, i.e., the pedal, lever, or a driving assistance program, sends a signal to the Electronic Stability Control (ESP)-ECU, which orchestrates the conventional and the regenerative brakes (C-Brake and R-Brake). The further relevant signals, such as the current speed, battery status, and friction are received via a communication gateway (GW) from the residual vehicle.

In addition to the presented parts, modern braking systems include several feedback signals, programs for driving stability, dynamics and motion management, effect chains, communication and failure paths as well as distance and error memories.

Although considering these things would emphasize the need for automation, they are omitted here for the sake of clarity.

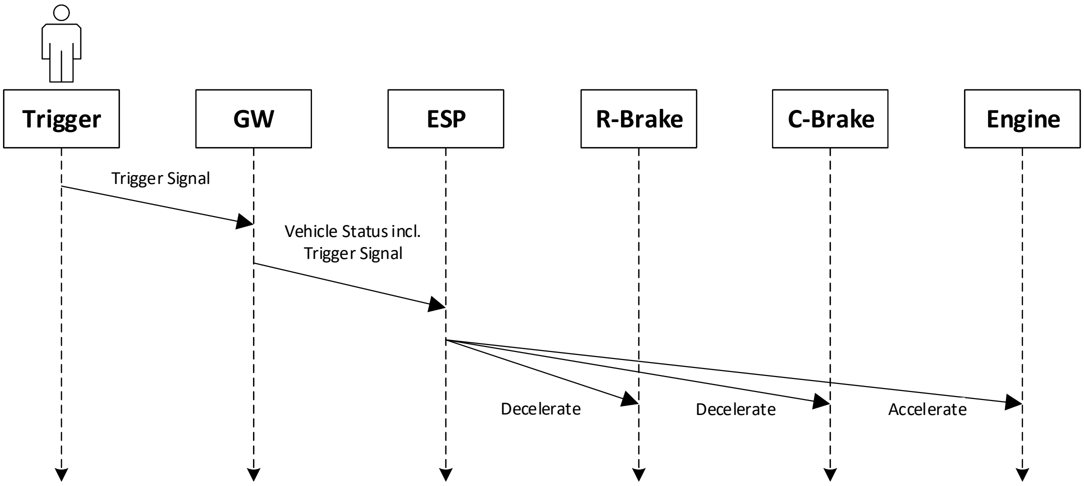


Figure 7.2: Use Case - Braking Sequence Diagram

Listing 7.1: Use Case - Basic System Model

```
# Components
C_GW  a Component ; label "Gateway"            ; stores D_GW-ESP ;
                                                 hosts F_GW  .
C_ESP a Component ; label "ESP ECU"            ; hosts F_ESP .
C_RB  a Component ; label "Regenerative Brake" ; hosts F_RB  .
C_CB  a Component ; label "Conventional Brake" ; hosts F_CB  .
C_PT  a Component ; label "Engine"             ; hosts F_PT  .

# Functions
F_GW  a  Source   ; produces  D_GW-ESP .
F_ESP a  Function ; consumes  D_GW-ESP ;
                    produces  D_ESP-CB, D_ESP-PT, D_ESP-RB .
F_RB  a  Sink     ; consumes  D_ESP-RB .
F_CB  a  Sink     ; consumes  D_ESP-CB .
F_PT  a  Sink     ; consumes  D_ESP-PT .

# Data
D_GW-ESP a  Data  ; label "Vehicle Status"      .
D_ESP-RB a  Data  ; label "Deceleration Torque" .
D_ESP-CB a  Data  ; label "Deceleration Torque" .
D_ESP-PT a  Data  ; label "Acceleration Torque" .

# Connections
N_GW-ESP a  Connection ; transmits D_GW-ESP ; connects  F_GW  , F_ESP .
N_ESP-RB a  Connection ; transmits D_ESP-RB ; connects  F_ESP , F_RB  .
N_ESP-CB a  Connection ; transmits D_ESP-CB ; connects  F_ESP , F_CB  .
N_ESP-PT a  Connection ; transmits D_ESP-PT ; connects  F_ESP , F_PT  .
```

7.2 Model of the System under Evaluation

As described in Section 4.2.4, the first step of an assessment is the development of a usable model from the available information. Using the provided SSML, the architecture of the SuE, shown in Figure 7.1, is described in Listing 7.1. To support clarity, the label properties in the listing refer to those used in the figure. The security-relevant extensions are subsequently derived and augmented by the impact and threat analyses.

7.3 Security Risk Assessment

In this section, the SRA process, specified in Chapter 4 is demonstrated. In line with the quest for a high degree of automation, the majority of the analysis is based on assessment catalogs for the relevant criteria and ratings.

7.3.1 Impact Assessment

The impact assessment involves the initial rating, propagation, and annotation of the CILs to the security goals, as specified in Section 4.3. Therefore, two tables are created from the basic system model. Table 7.1 describes the propagation of inputs to the SuE and their top-down propagating security objective confidentiality. The results of the bottom-up propagated security objectives, i.e. integrity, availability, authenticity, and non-repudiation, are then displayed in Table 7.2. The derived CILs are annotated to the corresponding security goals, with the initially provided ratings printed in bold. Since logging functions are not in the scope of this use case, the security objective non-repudiation is left unconsidered in the further assessments.

Table 7.1: Use Case - Top-Down Propagating Impact Ratings

Instance	Description	CIL_{Conf}	Vector	Rationale
F_{GW}	Routing Function	1	$\max(1, D_{\text{GW-ESP}})$	
C_{GW}	Gateway	1	F_{GW}	
$D_{\text{GW-ESP}}$	Residual Vehicle Data	**1**	Intrinsic	Personal data (speed, traction)
$N_{\text{GW-ESP}}$	Transmits $D_{\text{GW-ESP}}$ from F_{GW} to F_{ESP}	1	$D_{\text{GW-ESP}}$	
F_{ESP}	Calculates the individual commands from $D_{\text{GW-ESP}}$	**3**	Intrinsic:	Intellectual property
C_{ESP}	ESP ECU	3	F_{ESP}	
$D_{\text{ESP-RB}}$	Signal to regenerative brakes	3	F_{ESP}	Intellectual property
$N_{\text{ESP-RB}}$	Transmits $D_{\text{ESP-RB}}$ from F_{ESP} to F_{RB}	3	$D_{\text{ESP-RB}}$	
F_{RB}	Regenerative brakes function	3	$D_{\text{ESP-RB}}$	
C_{RB}	Regenerative brakes ECU	3	F_{RB}	
$D_{\text{ESP-CB}}$	Signal to conventional brakes	**0**	Intrinsic	No damage
$N_{\text{ESP-CB}}$	Transmits $D_{\text{ESP-CB}}$ from F_{ESP} to F_{CB}	0	$D_{\text{ESP-CB}}$	
F_{CB}	Conventional brakes function	0	$D_{\text{ESP-CB}}$	
C_{CB}	Conventional brakes ECU	0	F_{CB}	
$D_{\text{ESP-PT}}$	Signal to powertrain	**2**	Intrinsic	Intellectual property
$N_{\text{ESP-PT}}$	Transmits $D_{\text{ESP-PT}}$ from F_{ESP} to F_{PT}	2	$D_{\text{ESP-PT}}$	
F_{PT}	Engine Control Function	2	$D_{\text{ESP-PT}}$	
C_{PT}	Powertrain ECU	2	F_{PT}	

Table 7.2: Use Case - Bottom-Up Propagating Impact Ratings

Instance	$\mathrm{CIL}_{\mathrm{Int}}$	$\mathrm{CIL}_{\mathrm{Ava}}$	$\mathrm{CIL}_{\mathrm{Aut}}$	$\mathrm{CIL}_{\mathrm{NR}}$	Vector	Rationale
C_{GW}	4	3	4	2	F_{GW}	
F_{GW}	4	3	4	2	$D_{\mathrm{GW-ESP}}$	
$N_{\mathrm{GW-ESP}}$	4	3	4	2	$D_{\mathrm{GW-ESP}}$	
$D_{\mathrm{GW-ESP}}$	4	3	4	2	F_{ESP}	
C_{ESP}	4	3	4	2	F_{ESP}	
F_{ESP}	4	3	4	2	$\max(D_{\mathrm{ESP-RB}}, D_{\mathrm{ESP-CB}}, D_{\mathrm{ESP-PT}})$	
$N_{\mathrm{ESP-RB}}$	3	1	3	1	$D_{\mathrm{ESP-RB}}$	
$D_{\mathrm{ESP-RB}}$	3	1	3	1	F_{RB}	
C_{RB}	3	1	3	1	F_{RB}	
F_{RB}	**3**	**1**	**3**	**1**	Intrinsic	Aut: Self-braker Ava: R-Brake failure Int: Both NR: Forensic
$N_{\mathrm{ESP-CB}}$	3	3	3	2	$D_{\mathrm{ESP-CB}}$	
$D_{\mathrm{ESP-CB}}$	3	3	3	2	F_{CB}	
C_{CB}	3	3	3	2	F_{CB}	
F_{CB}	**3**	**3**	**3**	**2**	Intrinsic	Aut: Self-braker Ava: C-Brake failure Int: Both NR: Forensic
$N_{\mathrm{ESP-PT}}$	4	3	4	2	$D_{\mathrm{ESP-PT}}$	
$D_{\mathrm{ESP-PT}}$	4	3	4	2	F_{PT}	
C_{PT}	4	3	4	2	F_{PT}	
F_{PT}	**4**	**3**	**4**	**2**	Intrinsic	Aut: Self-accelerator Ava: Engine failure Int: Both N-R: Forensic

7.3.2 Threat Assessment

The threat assessment takes place after the impact assessment, as described in Section 4.4. It involves the assessment of the relevant attacker models, which define the expected PAPs, the derivation of the security goal-specific RAP, and the costs-benefits ratio from an attacker's point-of-view.

Attacker Model and Motivation

Deemed relevant attacker types for this use case are the vehicle owner, the professional tuner, the (remote-)saboteur, and the competitor. Their individual estimated PAP, based on historical data, expert knowledge, and ground truth, is shown in Table 7.3.

Table 7.3: Use Case - Attacker Model Specific Attack Potential

Attacker Model	Expertise	Knowledge	Equipment	Time	Type of Access
Vehicle Owner	Layman	Public	Standard	Days	Easy
Professional Tuner	Proficient	Restricted	Specialized	Weeks	Moderate
Competitor	Multiple experts	Restricted	Multiple bespoke	Months	Difficult
Saboteur	Proficient	Public	Specialized	Weeks	Remote

Attractive targets for the vehicle owner and the tuner are performance-tuning by access to acceleration reserved for the exclusive use by the ESP, and efficiency-tuning by shifting deceleration torque from conventional to regenerative braking, whereby the safety-relevant deceleration stability is sacrificed in favor of energy recovery. Both, the tuner and the owner intend to manipulate the regenerative brake and engine (powertrain) functionality by manipulation of components, connections, and data, or by injection of commands. The tuner shares the interests of the vehicle owner, but has considerably higher capabilities, including access to a fully equipped workshop where he can replace and manipulate components. Factors for apprehension and conviction are here not considered despite a serious risk of accidents in case of improper manipulation. The assumed monetary and psychological benefit of a vehicle owner by disclosing information about his braking signals is rather small, as the value is not the individual signals but the underlying control-algorithm. Assuming costs of several hundred Euros for the equipment, training, and time for the extraction, compared to a quantified psychological benefit of at best a thousand Euros, the probability is low, according to Equation 4.4 on page 59. The reverse-engineering required to make attractive use of this data (e.g., for in-vehicle displays) increases costs by several thousand Euros, reducing attractivity for the vehicle owner and tuner to a negligible value.

Reverse-engineering of the novel regenerative brake control-algorithm is of particular interest for the competitor attacker model, who intends to acquire valuable know-how for the improvement of his products. To this end, he makes considerable efforts, including the provision of specially equipped laboratories and experts in various fields.

The immobilization or impairment of the vehicle to the detriment of the vehicle owner is solely the intention of the saboteur attacker model. Therefore, he intends to affect the availability and integrity of the overall SuE. Since these intentions diametrically oppose the vehicle owner, the saboteur is limited to a remote, occasionally external access to the vehicle, while the owner, tuner, and competitor have unlimited physical access.

The motivation of the different attacker models to violate a security goal is indicated in Table 7.5 on the next page by bullets (•).

Required Attack Potential

The generic RAPs that an attacker has to expend to violate a security goal is taken from a prepared catalog, as shown in Table 7.4, and annotated to each corresponding security goal. In practice, these estimations strongly depend on the used technologies and should be refined according to the implementation details. The individual capability of each considered attacker model to impair a particular security goal of the SuE is visualized in Table 7.5 by squares (□).

Table 7.4: Use Case - Required Attack Potential per Threat and Instance Class

Threat	Instance Class	Expertise	Knowledge	Equipment	Access	Time	Example
Spoofing	Function	Layman	Public	Standard	Remote	Days	Manipulate input
	Component	Proficient	Public	Specialized	Moderate	Hours	Install manipulated or counterfeit component
	Data	Proficient	Public	Standard	Remote	Days	Manipulate stored or transmitted data
Tampering	Function	Expert	Restricted	Specialized	Easy	Hours	Software manipulation
	Component	Expert	Restricted	Specialized	Moderate	Days	Hardware manipulation
	Data	Proficient	Restricted	Standard	Remote	Days	Manipulate stored data
	Connection	Proficient	Restricted	Standard	Moderate	Hours	Manipulate data in transit
Repudiation	Function	Expert	Restricted	Standard	Remote	Hours	Manipulate logging
	Component	Proficient	Restricted	Specialized	Easy	Hours	Manipulate stored log
Information Disclosure	Function	Expert	Public	Specialized	Remote	Weeks	Software reverse-engineering
	Component	Expert	Public	Specialized	Moderate	Weeks	Hardware reverse-engineering
	Data	Layman	Public	Standard	Remote	Hours	Data extraction
	Connection	Layman	Public	Standard	Easy	Hours	Eavesdropping
Denial of Service	Function	Layman	Public	Standard	Remote	Hours	Disrupt computation
	Component	Layman	Public	Standard	Easy	Hours	Break component
	Data	Layman	Public	Standard	Remote	Hours	Delete or damage stored data
	Connection	Layman	Public	Standard	Easy	Hours	Disrupt connection
Elevation of Privilege	Function	Proficient	Restricted	Standard	Remote	Weeks	Unauthorized use of Functions
	Component	Proficient	Restricted	Standard	Remote	Weeks	Add functions

Table 7.5: Use Case - Attractive (•) and Capable (□) Security Objectives

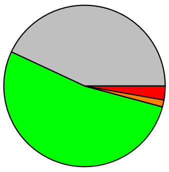
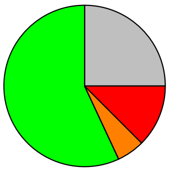
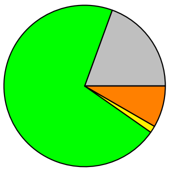
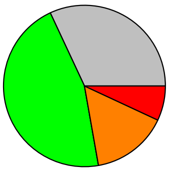
Risk Situation	Vehicle Owner				Tuner				Competitor				Saboteur			
Instance	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut
C_{GW}			□				□	⊡	□		□	□		•	•	
F_{GW}			□	□		•	□	⊡	□		□	□		•	⊡	⊡
$N_{\mathrm{GW-ESP}}$	□	•	□		□	•	□		□	□	□			•	•	
$D_{\mathrm{GW-ESP}}$	□		□		□	⊡	□	□	□	□	□	□	□	•	⊡	⊡
C_{ESP}			□			•	□	⊡	⊡		□	□		•	•	
F_{ESP}		•	□	⊡		•	□	⊡	⊡		□	□		•	⊡	⊡
$N_{\mathrm{ESP-RB}}$	□	•	□		□	•	□		⊡	□	□			•	•	
$D_{\mathrm{ESP-RB}}$	□	•	□	•	□	⊡	□	⊡	⊡	□	□	□	□	•	⊡	⊡
C_{RB}			□			•	□	⊡	⊡		□	□		•	•	
F_{RB}		•	□	⊡		•	□	⊡	⊡		□	□		•	⊡	⊡
$N_{\mathrm{ESP-CB}}$	□		□		□		□		□	□	□			•	•	
$D_{\mathrm{ESP-CB}}$	□		□		□	□	□	□	□	□	□	□	□	•	⊡	⊡
C_{CB}			□				□	□	□		□	□		•	•	
F_{CB}			□	□			□	□	□		□	□		•	⊡	⊡
$N_{\mathrm{ESP-PT}}$	□	•	□		□	•	□		□	□	□			•	•	
$D_{\mathrm{ESP-PT}}$	□	•	□	•	□	⊡	□	⊡	⊡	□	□	□	□	•	⊡	⊡
C_{PT}			□			•	□	⊡	□		□	□		•	•	
F_{PT}		•	□	⊡		•	□	⊡	□		□	□		•	⊡	⊡

7.3.3 Risk Calculation

The individual risk to each security goal is calculated from the probability and impact values, as described in Section 4.5, and shown in Table 7.6. The capable but unattractive risks are considered to be low since there is always the possibility that an attacker might inadvertently execute even unattractive threats. Materializing these risks by carrying out the relevant threats is comparatively simple. To reach their goals, the vehicle owner and tuner simply have to inject their own commands for the engine and the brake control function into the vehicle bus network. The competitor could simply download the functions and study the components and signals to learn about the functionality. As practically all modern vehicles have at least one remote connection, also remote manipulation is possible by the saboteur.

To reduce the identified risks, some of which are very high, appropriate threat mitigation is necessary. A demonstration of the methods introduced for this purpose in Chapter 5 is the subject of the following section.

Table 7.6: Use Case - Security Risks (unmitigated)

Risk Situation	Vehicle Owner				Tuner				Competitor				Saboteur			
Instance	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut
C_{GW}			Low				Low	Very high	Low		Low	Low		Low	Low	
F_{GW}			Low	Low		Low	Low	Very high	Low		Low	Low		Low	High	Very high
$N_{\mathrm{GW-ESP}}$	Low	Low	Low		Low	Low	Low		Low	Low	Low			Low	Low	
$D_{\mathrm{GW-ESP}}$	Low		Low		Low	Very high	Low	Low	Low	Low	Low	Low	Low	Low	High	Very high
C_{ESP}			Low			Low	Low	Very high	High		Low	Low		Low	Low	
F_{ESP}		Low	Low	Very high		Low	Low	Very high	High		Low	Low		Low	High	Very high
$N_{\mathrm{ESP-RB}}$	Low	Low	Low		Low	Low	Low		High	Low	Low			Low	Low	
$D_{\mathrm{ESP-RB}}$	Low	Low	Low	Low	Low	High	Low	High	High	Low	Low	Low	Low	Low	Low	High
C_{RB}			Low			Low	Low	High	High		Low	Low		Low	Low	
F_{RB}		Low	Low	High		Low	Low	High	High		Low	Low		Low	Low	High
$N_{\mathrm{ESP-CB}}$	Low		Low		Low		Low		Low	Low	Low			Low	Low	
$D_{\mathrm{ESP-CB}}$	Low		Low		Low	Low	Low	Low	Low	Low	Low	Low	Low	Low	High	High
C_{CB}			Low				Low	Low	Low		Low	Low		Low	Low	
F_{CB}			Low	Low			Low	Low	Low		Low	Low		Low	High	High
$N_{\mathrm{ESP-PT}}$	Low	Low	Low		Low	Low	Low		Low	Low	Low			Low	Low	
$D_{\mathrm{ESP-PT}}$	Low	Low	Low	Low	Low	Very high	Low	Very high	Moderate	Low	Low	Low	Low	Low	High	Very high
C_{PT}			Low			Low	Low	Very high	Low		Low	Low		Low	Low	
F_{PT}		Low	Low	Very high		Low	Low	Very high	Low		Low	Low		Low	High	Very high
□ No 🟩 Low 🟨 Moderate 🟧 High 🟥 Very high																

7.4 Derivation and Management of Security Requirements

This section encompasses the derivation of intrinsic security requirements, as described in Section 5.3, under consideration of an extrinsic requirement, as described in section 5.4.

7.4.1 Extrinsic Security Requirements

The example of the extrinsic security requirement of this use case is a regulation that requires brake signals to be authenticated, while encryption is prohibited. According to the unified security requirements data model, described in Section 5.4, in particular Figure 5.3 on page 71, this translates to an extrinsic security requirement for the functional security requirement authentication, that demands use of a null cipher suite [153] for the relevant connection.

7.4.2 Intrinsic Security Requirements

The risks, previously shown in Table 7.6, implicitly indicate the need for proper mitigation at an abstract level, such as the integrity of $D_{\text{ESP-RB}}$ must be protected due to a high risk. To refine these abstract requirements in an actionable way, security controls from a catalog are recommended and presented in Table 7.7. Specific dependencies that need to be considered when implementing the recommended security controls are indicated. For this use case, these are Secure Key Management (KM), Secure Implementation (SI) and Secure Network Configuration (SNC).

To reduce the remaining risks for the confidentiality of data and connections to an acceptable level, either encryption or access control can be used to increase the RAP.

With regard to the extrinsic requirement, the encryption-based security controls are not applicable. This leads to the less effective security controls of network segmentation and access control for stored data.

Performing this process for the different attacker models, based on their individual SRA, allows the establishment of some overlapping security controls, providing an appropriate level of security with a minimized residual risk.

Table 7.7: Use Case - Recommended Security Controls

Security Control	Objective	Instance Class	Affected Factor	Minimum Effort	Dependency
Cryptographic Signature	Int, Aut	Function, Data	Expertise Time	Expert Years	SI, KM
Network Segmentation	Conf, Int, Aut	Function, Data, Connection	Expertise Type of Access	Proficient Moderate	SNC
	Ava	Function, Data, Component	Type of Access	Easy	
HSM	Aut	Component	Expertise Time	Expert Years	SI, KM
Hardening (HW)	Conf, Int	Component	Expertise Equipment Time Type of Access	Expert Bespoke Months Hard	SI
Hardening (SW)	Conf, Int	Function	Expertise Time	Expert Years	SI, KM

The recommended security controls for this use case are introduced below.

Cryptographic Signatures are an efficient security control to prevent tampering of functions by manipulated and injected input data. As most data and connections in this use case inherit their damage from their destined function (except for confidentiality), their risks are also mitigated by this control.

Network Segmentation prevents unauthorized access to sensitive parts of the vehicle network by virtually dividing it into separate networks between which only predefined signals are routed. Here, this control prevents the transmission of remote signals from other segments, like infotainment, to the safety-relevant brake segment. For this use case, it further includes some basic physical protection by making the communication lines hardly accessible, e.g., inside a reinforced chassis frame.

Hardware Security Module (HSM) is a physical device as a part of a component that provides cryptographic functionalities, including the storage of key materials in a secure way. By effective prevention of private key material extraction, it guarantees the authenticity of a component.

Software Hardening is, in the scope of this use case, an aggregation of various methods to prevent manipulation and analysis of functions. It includes in particular secure boot, encapsulation, runtime monitoring, and prevention of unauthorized updates.

Hardware Hardening aggregates in this use case measures to protect components from tampering and information disclosure. This includes, among others, closed debug interfaces, and physical tamper protection.

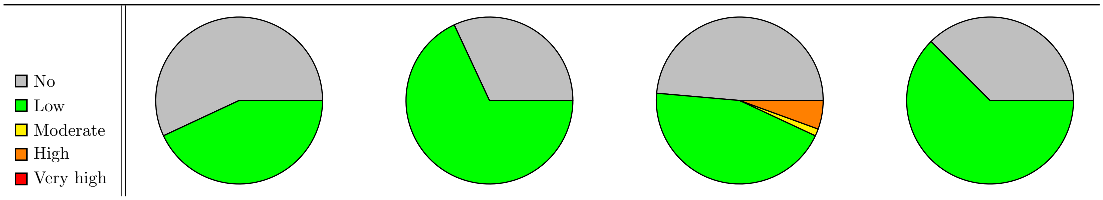
By these security controls, the capable threats list is updated, as shown in Table 7.8, which leads to the new risk situation, shown in Table 7.9.

Table 7.8: Use Case - Updated Attractive (•) and Capable (□) Security Objectives

Risk Situation	Vehicle Owner				Tuner				Competitor				Saboteur			
Instance	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut
C_{GW}			□				□	•	□		□			•	•	
F_{GW}			□			•	□	•			□			•	•	•
$N_{\text{GW-ESP}}$		•	□		□	•	□		□	□	□			•	•	
$D_{\text{GW-ESP}}$			□		□	•	□		□		□			•	•	•
C_{ESP}			□			•	□	•	⊡		□			•	•	
F_{ESP}		•	□	•		•	□	•	•		□			•	•	•
$N_{\text{ESP-RB}}$		•	□		□	•	□		⊡	□	□			•	•	
$D_{\text{ESP-RB}}$		•	□	•	□	•	□	•	⊡		□			•	•	•
C_{RB}			□			•	□	•	⊡		□			•	•	
F_{RB}		•	□	•		•	□	•	•		□			•	•	•
$N_{\text{ESP-CB}}$			□		□		□		□	□	□			•	•	
$D_{\text{ESP-CB}}$			□		□		□		□		□			•	•	•
C_{CB}			□				□		□		□			•	•	
F_{CB}			□				□				□			•	•	•
$N_{\text{ESP-PT}}$		•	□		□	•	□		□	□	□			•	•	
$D_{\text{ESP-PT}}$		•	□	•	□	•	□	•	⊡		□			•	•	•
C_{PT}			□			•	□	•	□		□			•	•	
F_{PT}		•	□	•		•	□	•			□			•	•	•

Table 7.9: Use Case - Security Risks (mitigated)

Risk Situation	Vehicle Owner				Tuner				Competitor				Saboteur			
Instance	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut	Conf	Int	Ava	Aut
C_{GW}			Low				Low	Low	Low		Low			Low	Low	
F_{GW}			Low			Low	Low	Low			Low			Low	Low	Low
$N_{\mathrm{GW-ESP}}$		Low	Low		Low	Low	Low		Low	Low	Low			Low	Low	
$D_{\mathrm{GW-ESP}}$			Low		Low	Low	Low		Low		Low			Low	Low	Low
C_{ESP}			Low			Low	Low	Low	High		Low			Low	Low	
F_{ESP}		Low	Low	Low		Low	Low	Low	Low		Low			Low	Low	Low
$N_{\mathrm{ESP-RB}}$		Low	Low		Low	Low	Low		High	Low	Low			Low	Low	
$D_{\mathrm{ESP-RB}}$		Low	Low	Low	Low	Low	Low	Low	High		Low			Low	Low	Low
C_{RB}			Low			Low	Low	Low	High		Low			Low	Low	
F_{RB}		Low	Low	Low		Low	Low	Low	Low		Low			Low	Low	Low
$N_{\mathrm{ESP-CB}}$			Low		Low		Low		Low	Low	Low			Low	Low	
$D_{\mathrm{ESP-CB}}$			Low		Low		Low		Low		Low			Low	Low	Low
C_{CB}			Low				Low		Low		Low			Low	Low	
F_{CB}			Low				Low				Low			Low	Low	Low
$N_{\mathrm{ESP-PT}}$		Low	Low		Low	Low	Low		Low	Low	Low			Low	Low	
$D_{\mathrm{ESP-PT}}$		Low	Low	Low	Low	Low	Low	Low	Moderate		Low			Low	Low	Low
C_{PT}			Low			Low	Low	Low	Low		Low			Low	Low	
F_{PT}		Low	Low	Low		Low	Low	Low			Low			Low	Low	Low



7.4.3 Reflection of Vulnerabilities into the Model

The reflection of incidents into the model, as described in Section 6.4, is illustrated by means of two actually occurred vulnerabilities that each required updates of existing SRAs.

The first vulnerability is a broken signature validity check that reported all provided signatures as valid. The reason for this flaw was a mistranslation of the requirement "The signature validation must confirm the validity of the signature before processing of the data" by the supplier as "The signature validation must report valid before it processes the data", which led to an implementation of the *checkSignature* function call that simply always returned *TRUE*.

The second vulnerability was introduced by a widely used aftermarket ECU, which retrofits features like remote diagnosis, climate control, unlocking, ignition, etc. via a smartphone app. Therefore, the ECU must be connected to several vehicle bus segments. Unfortunately, the interface to the smartphone app was implemented as an insecure http connection that allows direct read and write access to the connected bus segments, thus removing the protective effects of network segmentation.

Table 7.10: Use Case - Vulnerabilities Mapping

Vulnerability	Mitigates Control	Objective	Target	Affected Factor	Effort Old	Effort New
Broken Signature Validation	Cryptographic Signature	Int, Aut	Function, Data	Expertise Time	Expert Years	*Original *Original
Insecure Remote Interface	Network Segmentation	Int, Ava Aut	Function, Data, Connection	Expertise Type of Access	Proficient Moderate	*Original Remote
		Conf	Component	Type of Access	Easy	*Original

8 Conclusion and Future Work

In this final chapter, the previous chapters' results are summarized by a discussion of the initially formulated research questions, followed by a general conclusion. Turning to the future, an outlook is given on how the provided contributions can be continued and improved by future work.

8.1 Discussion of Answers to the Research Questions

The overall objective of this work is the sustainable increase of CPS security. Therefore, security risk assessment and requirements management have to be automated wherever possible. Answering the research questions raised in the course of this research led to a semi-automatic workflow with new and significantly improved methods that strongly support security risk management throughout the CPS lifecycle.

The first question **RQ 1** on **"How can potential security risks for Cyber-Physical Systems be systematically identified and assessed, in an automated way?"** is approached by a model-based security risk assessment. A vital point for the successful identification and assessment of security risk is the tight cooperation of security analysts and domain experts. Jointly creation of organization- and domain-specific catalogs and the transformation of existing descriptions into machine-readable models enable the automation of security risk assessment. An appropriate model for this purpose is presented in Section 4.2.4, followed by a graph-based risk assessment methodology that enables the individual assessment of risks for each security goal. For simple cases, the modeling may appear exaggerated. However, with a high number of individual instances in complex CPSs, the semi-automated approach has clear advantages. While the creation of the basic system model and the assessment of potential impacts to sinks, sources, and special cases require human effort, large parts of the analysis can be efficiently automated by the methods described in Chapter 4.

The second question **RQ 2** reads as **"How to automatically formulate security requirements that support the achievement of the indicated level of protection?"**. Answering this question implies questions about the correct formulation of security requirements and for the derivation of appropriate security requirements. A hierarchic taxonomy and machine-readable data model for the formulation are presented in Section 5.2. Their successful implementation and insights from the evaluation by several industrial partners are described in Section 5.2.4. The process to automatically synthesize appropriate intrinsic security requirements from the identified risks by a pattern-based mapping of threat mitigations to the security goals at risk is described in Section 5.3.

The answer to the third question **RQ 3**, **"How to match extrinsic security requirements with the provided security capabilities?"** also benefits from the introduced taxonomy. The uniform formulation of extrinsic and intrinsic security requirements enables the automated validation of compliance between requested and implemented capabilities, as described in Section 5.4. A still open challenge is the translation of the natural language extrinsic requirements into the data model. Successful automation of this task requires in-depth practical experience of manual translation and validation of automated translations into the data model.

The fourth question **RQ 4**, **"How to detect, analyze and respond to potentially disruptive changes of the security situation?"**, calls for proper incident detection, correlation, and management. Approaching these calls locally, on the one hand, a CPS-specific method to record and forward network data for the detection of local incidents is introduced in Section 6.2. On the other hand, large-scale attack campaigns against distributed supply chains are dealt with by a method to classify and exchange indicators of compromise in Section 6.3. The consideration of changing risk situations is achieved by means of targeted updates of the system and threat models, which is described in Section 6.4.

8.2 Conclusion

The work at hand is motivated by the need for appropriate security management for long-lasting CPSs. As initially stated, the mitigation and reduction of risks caused by malicious activities are subject to constant change. Appropriate situational awareness requires knowledge of current risks, their mitigation, and the ability to recognize the signs of the times in order to make timely adjustments. For this very reason, the main objective of this research was to improve the general security level of CPSs in the field by automating complex and time-consuming process steps.

The presented semi-automated methods and processes are the result of conducting more than a hundred security risk assessments and security concept development projects in various fields, with a strong focus on the automotive sector (cf. Section 2.6). The preceding chapters show how a high degree of automation of security risk assessment and requirements management is achievable in practice.

The respective degree of automation strongly depends on whether individual assessments and measures are required, or catalogs of standardized assessments and solutions can be reused. Their applicability was demonstrated by a simplified use case, which is based on a real-world implementation.

The presented framework accordingly comprises three main components: A model-based security risk assessment methodology, methods to unify, deduce and manage security requirements, and a set of tools and procedures to detect and respond to security-relevant situations. The SRA part includes a semantic meta-model, by which the SuE can be described as a machine-readable model. Based on such a model, a graph-based analysis is performed, considering several factors including the relevant threats, their expected impact, the attacker's motivation and potential, and protective measures. The work product of this process is an enumeration of rated security risks. Compared to most other security risk assessment processes, the method is clearly structured and supports a high degree of automation. Based on the enumerated security risks, intrinsic security requirements protecting the system and its intended functionality can be deduced. For automatic processing and validation of such requirements, an object-oriented security requirements data model is introduced. By such structured security requirements, it becomes possible to validate automatically if existing structures match with new extrinsic requirements. Upcoming IIoT use cases, such as identifying suitable production lines for security-critical processes that require specific security capabilities, are successfully evaluated. Knowing the potential risks and appropriate security measures in place, monitoring their validity is the next step towards long-term security.

For an appropriate response, threats must be identified, assessed, and correlated where necessary. If the overall situation changes, it must be re-assessed and, if required, corrective action must be taken. A common way to detect attacks on CPSs is anomaly detection. As CPS are typically constrained in their capabilities, this detection is done by a remote anomaly detection system that monitors the network communication. The substantial amounts and the confidentiality of the transmitted data often prevent them from being forwarded to such anomaly detection systems. To counter this situation, a bandwidth- and privacy-preserving protocol for M2M-communication was introduced, supporting network traffic anomaly detection by various systems. While network anomaly detection and analysis is an exciting, own field of research, it is not in the scope of this work. Information about detected incidents must be assessed and ideally exchanged with further potential targets. These can be within but also outside the own organization. The process presented to analyze, categorize, report and correlate incident information combines machine learning with semantic reasoning to enable selective sharing of IoCs, which is, in particular, necessary and common for critical infrastructures, to prevent market distortions and to correlate large-scale attack campaigns. Since threat information such as vulnerabilities or incidents means a change in the overall security situation, the final point on this topic is to update the model, initiating a further evaluation of the security situation. Based on the updated security risk assessment, the already implemented security capabilities can be matched with new requirements.

Unified catalogs of security risk mitigations are necessary, while catalogs and rating schematics for the assessment of impacts should be individual for the domains and organizations. The manifestation of security as a cross-sectional discipline is long overdue, given the severe influences of critical CPSs on modern life. As digitization progresses, the need for practical, systematic, and justified security requirements is growing.

The presented risk assessment and requirements generation methods were evaluated by a practical example. In the absence of suitable evaluation criteria for the quality of analyses and generated requirements, 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.

8.3 Future Work

Drawing on the work presented and a couple of decisions that might be challenged, there are plenty of opportunities for further improvements waiting to be explored.

The presented graph-based models provide an intuitive and straightforward structure that has been used for practical modeling and analysis of SuE. Nevertheless, creating an initial model remains a complicated, error-prone, manual task that should benefit from further automation. So far, the focus has been on the methodology of risk assessment and risk management. The addition of model checking techniques and axioms, which ensure the consistency of the created models and prevent flaws, are undoubtedly useful. In favor of further automation to improve efficiency, an appropriate balance must be found between increased complexity, modeling confidence, and practical applicability.

Another critical issue is the further meaningful use of the work products of the security risk assessments and requirements generation as input for downstream quality management processes, like automated test management and situational anomaly detection.

Attacks following the cyber-kill-chain [154] will increase, as CPSs keep becoming strategic targets. One of the primary reasons for overhead in security risk management is the lack of consolidated and structured methodologies among the different application domains. While security implementation catalogs enable an automated implementation of derived security requirements, they must continuously be updated and extended. While harmonization of threat mitigation techniques helps to establish a consistent minimum level of security, off-the-shelf protection also supports advanced attackers, as they have to develop fewer different exploits. The diversification of CPSs and their protection is so far one of the most remarkable factors in the prevention of large-scale attacks, which is threatened by ongoing harmonization. Currently, there is a lack of techniques to simultaneous standardize security implementations while tailoring them to the specific characteristics of individual CPS networks.

A further future research topic is the processing of the presented methodologies into small and medium-sized enterprises. The catalogs for model-based methods presented in this thesis have so far been primarily aimed at large organizations that can create and maintain such catalogs themselves. Small manufacturers and operators usually lack the capacity to establish such methods in their daily business. Standardized catalogs may facilitate applicability in this area but lead to the vulnerable monocultures mentioned above.

However, the initially pointed out paradigm shift in the CPS domain from dedicated, insecure fieldbuses to more dynamic, service-oriented architectures is a rare opportunity to implement reliable and maintainable security from the very beginning. Although it will take decades for the current dark shadows over the insecure CPSs to dissolve, risk-based security management helps to reduce the feasibility and damage of cyberattacks in the long run.

Bibliography

[1] C. Miller and C. Valasek. "Adventures in automotive networks and control units." In: *Def Con* 21 (Aug. 2013), pp. 260–264.

[2] F. D. Garcia et al. "Lock it and still lose it—on the (in) security of automotive remote keyless entry systems." In: *25th USENIX Security Symposium*. USENIX Security '16. Austin, TX, USA, Aug. 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/garcia (visited on 2020-07-26).

[3] BSI. *The State of IT Security in Germany 2018*. German. Report. Bonn, Germany: Federal Office for Information Security (BSI), Sept. 2018. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2018.pdf (visited on 2020-07-26).

[4] K. E. Hemsley and R. E. Fisher. *History of Industrial Control System Cyber Incidents*. Technical Report. Idaho Falls, ID, USA: Idaho National Lab.(INL), Dec. 2018.

[5] BSI. *The State of IT Security in Germany 2019*. German. Report. Bonn, Germany: Federal Office for Information Security (BSI), Oct. 2019. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2019.pdf (visited on 2020-07-26).

[6] S. Adepu et al. "Investigation of Cyber Attacks on a Water Distribution System." In: *arXiv preprint arXiv:1906.02279* abs/1906.02279 (June 2019). arXiv: 1906.02279. URL: http://arxiv.org/abs/1906.02279.

[7] G. Liang et al. "The 2015 Ukraine Blackout: Implications for False Data Injection Attacks." In: *IEEE Transactions on Power Systems* 32.4 (July 2016), pp. 3317–3318. ISSN: 0885-8950. DOI: 10.1109/TPWRS.2016.2631891.

[8] Z. Durumeric et al. "The Matter of Heartbleed." In: *Proceedings of the 2014 Conference on Internet Measurement Conference*. IMC '14. Vancouver, BC, Canada: Association for Computing Machinery, Nov. 2014, pp. 475–488. ISBN: 978-1-4503-3213-2. DOI: 10.1145/2663716.2663755.

[9] P. Kocher et al. "Spectre attacks: Exploiting speculative execution." In: *arXiv preprint arXiv:1801.01203* (Jan. 2018), pp. 1–16.

[10] M. Vanhoef and E. Ronen. "Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd." In: *IEEE Symposium on Security & Privacy (SP)*. Piscataway, NJ, USA: IEEE, 2020.

[11] P. Lünnemann and M. Wuschke. *Volkswagen Infotainment Web Interface protocol specification (viwi protocol)*. Member Submission. W3C, July 2019. URL: https://www.w3.org/Submission/viwi-protocol (visited on 2019-10-20).

[12] International Electrotechnical Commission and others. *IEC TS 62443-1-1:2009, Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models*. Standard. Geneva, CH: IEC, July 2009.

[13] International Electrotechnical Commission and others. *IEC 62443-3-3:2013, Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels*. Standard. Geneva, CH: IEC, Aug. 2013.

[14] International Electrotechnical Commission and others. *IEC 62443-4-2:2019 Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components*. Standard. Geneva, CH: IEC, Feb. 2019.

[15] Working Party on Automated/autonomous and Connected Vehicles. *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), June 2020. URL: http://www.unece.org/fileadmin/DAM/trans/doc/2020/wp29grva/ECE-TRANS-WP29-2020-079-Revised.pdf (visited on 2020-07-26).

[16] International Organization for Standardization. *ISO/SAE DIS 21434 Road Vehicles – Cybersecurity engineering*. Draft International Standard. Geneva, CH: ISO/SAE, 2020.

[17] G. Hansch, P. Schneider, and G. S. Brost. "Deriving Impact-driven Security Requirements and Monitoring Measures for Industrial IoT." In: *5th ACM Cyber-Physical System Security Workshop*. CPSS '19. Auckland, New Zealand: Association for Computing Machinery, July 2019, pp. 37–45. DOI: `10.1145/3327961.3329528`.

[18] Fraunhofer-Institut für Angewandte und Integrierte Sicherheit (AISEC). *IUNO Projekt*. URL: `https://iuno-projekt.de` (visited on 2020-07-26).

[19] D. Angermeier et al. "Modeling Security Risk Assessments." In: *17th escar Europe – Embedded Security in Cars*. ESCAR '19. Stuttgart, Germany: Ruhr-Universität Bochum, Universitätsbibliothek, Oct. 2019, pp. 133–146. DOI: `10.13154/294-6670`.

[20] G. Hansch et al. "A Unified Architecture for Industrial IoT Security Requirements in Open Platform Communications." In: *24th IEEE Conference on Emerging Technologies and Factory Automation*. ETFA '19. Zaragoza, Spain: IEEE, Sept. 2019, pp. 325–332. DOI: `10.1109/ETFA.2019.8869524`.

[21] S. Plaga et al. "Secure your SSH Keys! Motivation and Practical Implementation of a HSM-based Approach Securing Private SSH-Keys." In: *17th European Conference on Cyber Warfare and Security*. ECCWS '18. Oslo, Norway: Academic Conferences International Limited, June 2018, pp. 370–379. ISBN: 978-1-911218-86-9.

[22] G. Hansch, P. Schneider, and S. Plaga. "Packet-wise Compression and Forwarding of Industrial Network Captures." In: *9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (University "Politehnica" of Bucharest, Romania). IDAACS '17. Bucharest, Romania: IEEE, Sept. 2017, pp. 66–70. ISBN: 978-1-5386-0696-4. DOI: `10.1109/IDAACS.2017.8095051`.

[23] J. Wolf et al. "Adaptive Modelling for Security Analysis of Networked Control Systems." In: *4th International Symposium for ICS & SCADA Cyber Security Research* (Queen's Belfast University, UK). ICS-CSR '16. Belfast, UK: BCS Learning & Development, Aug. 2016, pp. 64–73. DOI: 10.14236/ewic/ICS2016.8.

[24] G. Hansch, M. Hutle, and W. Fitzgerald. *Smart Grid Threat Analysis using an Attack Tree and Semantic Threat Graph Hybrid.* Poster presented at Workshop on European Smart Grid Cybersecurity: Emerging Threats and Countermeasures. Belfast, UK, Aug. 2016.

[25] B. Schneier. "Attack trees." In: *Dr. Dobb's journal* 24.12 (1999), pp. 21–29.

[26] W. M. Fitzgerald. "An Ontology Engineering Approach To Network Access Control Configuration." PhD thesis. Cork, Ireland: University College, Aug. 2010.

[27] K. Böttinger, G. Hansch, and B. Filipovic. "Detecting and Correlating Supranational Threats for Critical Infrastructures." In: *15th European Conference on Cyber Warfare and Security*. ECCWS '16. Munich, Germany: Academic Conferences International Limited, July 2016, pp. 34–41. ISBN: 978-1-910810-93-4.

[28] P. Wagner et al. "Applicability of Security Standards for Operational Technology by SMEs and Large Enterprises." In: *25th IEEE International Conference on Emerging Technologies and Factory Automation*. ETFA '20. Vienna, Austria: IEEE, Sept. 2020, pp. 1544–1551. DOI: 10.1109/ETFA46521.2020.9212126.

[29] M. Hutle et al. *D2.2 Threat and Risk Assessment Methodology*. Deliverable. SPARKS Consortium, Sept. 2015.

[30] H. Sandberg et al. *D2.3 Tools for Smart Grid Cyber Security*. Deliverable. SPARKS Consortium, Mar. 2016.

[31] R. Chabukswar et al. *D2.6 Smart Grid Vulnerability and Risk Assessment*. Deliverable. SPARKS Consortium, Mar. 2016.

[32] M. Howard and S. Lipner. *The Security Development Lifecycle*. Redmond, WA, USA: Microsoft Press, 2006. ISBN: 0735622140.

[33] R. Mueller and L. Bruno. *The Security Development LifeCycle*. Jan. 2012. URL: https://social.technet.microsoft.com/wiki/contents/articles/7100.the-security-development-lifecycle.aspx (visited on 2020-07-26).

[34] *ZVEI Industrie-Services – Lebenszyklus von Maschinen und Anlagen*. Leitfaden. Zentralverband Elektrotechnik- und Elektronikindustrie e.V, Mar. 2015. URL: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2015/april/ZVEI-Leitfaden_Industrie-Services/ZVEI-Leitfaden-Industrie-Services.pdf (visited on 2020-07-26).

[35] K. Stouffer et al. *SP 800-82 Rev. 2: Guide to Industrial Control Systems (ICS) Security*. Standard. Gaithersburg, USA: National Institute of Standards and Technology (NIST), May 2015. DOI: 10.6028/NIST.SP.800-82r2.

[36] M. Wang, J. A. Vandermaar, and K. D. Srivastava. "Review of condition assessment of power transformers in service." In: *IEEE Electrical Insulation Magazine* 18.6 (Nov. 2002), pp. 12–25. ISSN: 1558-4402. DOI: 10.1109/MEI.2002.1161455.

[37] R. Ross, M. McEvilley, and J. Carrier-Oren. *SP 800-160 Volume 1: Systems Security Engineering*. Standard. National Institute of Standards and Technology (NIST), Nov. 2016. DOI: 10.6028/NIST.SP.800-160v1.

[38] International Organization for Standardization. *ISO/IEC 27000:2018 Information technology – Security techniques – Information security management systems – Overview and vocabulary*. Standard. Geneva, CH: ISO, Feb. 2018.

[39] International Organization for Standardization. *ISO/IEC 27005:2018 Information technology – Security techniques – Information security risk management*. Standard. Geneva, CH: ISO, July 2018.

[40] International Organization for Standardization. *ISO 31000:2018 Risk management – Guidelines*. Standard. Geneva, CH: ISO, Feb. 2018.

[41] Common Criteria. *Common Methodology for Information Technology Security Evaluation: Evaluation Methodology*. Standard. Common Criteria, Apr. 2017. URL: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf (visited on 2020-07-26).

[42] Joint Task Force Transformation Initiative and others. *SP 800-30 Rev. 1: Guide for conducting risk assessments*. Standard. National Institute of Standards and Technology (NIST), Sept. 2012.

[43] MITRE. *Common Attack Pattern Enumeration and Classification (CAPEC)*. URL: `https://capec.mitre.org/` (visited on 2020-07-26).

[44] NIST. *National Vulnerability Database*. URL: `https://nvd.nist.gov/vuln` (visited on 2020-07-26).

[45] Symantec Corporate Offices. *SecurityFocus Vulnerability Database*. URL: `https://www.securityfocus.com/vulnerabilities` (visited on 2020-07-26).

[46] MITRE. *Common Weakness Enumeration (CWE)*. URL: `https://cwe.mitre.org/` (visited on 2020-07-26).

[47] R. A. Martin, S. M. Christey, and J. Jarzombek. “The Case for Common Flaw Enumeration.” In: *Proceedings of Workshop on Software Security Assurance Tools, Techniques, and Metrics*. SAMATE ’05 500-265. Gaithersburg MD 20899: National Institute of Standards & Technology (NIST), Feb. 2006, pp. 29–35.

[48] K. Knorr et al. “Automatisierung von Penetrationstest-Berichten mittels CWE.” In: *DACH Security 2011* (Sept. 2011), pp. 124–135. URL: `https://www.hochschule-trier.de/fileadmin/Hauptcampus/Fachbereich_Informatik/Personen/Konstantin_Knorr/Publikationen/Automatisierung_Pentests_v099-kk.pdf` (visited on 2020-07-26).

[49] NIST. *Common Platform Enumeration (CPE) Dictionary*. URL: `https://nvd.nist.gov/products/cpe` (visited on 2020-07-26).

[50] Cornell University. *arXiv.org e-Print archive*. URL: `https://arxiv.org/` (visited on 2020-07-26).

[51] The Open Web Application Security Project. *OWASP Application Security Verification Standard*. URL: `https://owasp.org/www-project-application-security-verification-standard/` (visited on 2020-07-26).

[52] The Open Web Application Security Project. *OWASP Proactive Controls*. URL: https://owasp.org/www-project-proactive-controls/ (visited on 2020-07-26).

[53] K. Böttinger. "Fuzzing with Stochastic Feedback Processes." Dissertation. Munich, Germany: Technische Universität München, Jan. 2019.

[54] M. A. Schneider et al. "Improving Security Testing with Usage-Based Fuzz Testing." In: *Risk Assessment and Risk-Driven Testing*. Ed. by F. Seehusen et al. Cham: Springer International Publishing, June 2015, pp. 110–119. ISBN: 978-3-319-26416-5.

[55] BSI. *IT-Grundschutz Catalogues*. Standard. Federal Office for Information Security (BSI), Dec. 2016. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/International/GSK_15_EL_EN_Draft.html (visited on 2020-07-26).

[56] ENISA. *Risk Management Glossary*. Attiki, Greece, 2019. URL: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary (visited on 2020-07-26).

[57] J. R. Clark and W. L. Davis. "A Human Capital Perspective On Criminal Careers." In: *Journal of Applied Business Research* 11.3 (1995), p. 58.

[58] J. Downie. *Hacking For Fun More Than Profit*. New York, NY, USA, June 2011. URL: https://web.archive.org/web/20110616153809/http://www.tnr.com/blog/the-study/89997/hacking-fun-more-profit (visited on 2020-07-26).

[59] D. Dolev and A. C. Yao. "On the Security of Public Key Protocols." In: *IEEE Transactions on Information Theory* 29.2 (Mar. 1983), pp. 198–208. ISSN: 0018-9448. DOI: 10.1109/TIT.1983.1056650.

[60] Communications-Electronic Security Group (UK). *HMG IA Standard Numbers 1 & 2 – Information Risk Management*. Standard. Withdrawn 2014. Gloucestershire, UK: CESG, Apr. 2012.

[61] International Electrotechnical Commission and others. *IEC 62443-4-2:2019 Security for industrial automation and control systems - Part 4-1: Secure product development lifecycle requirements*. Standard. Geneva, CH: IEC, Jan. 2018.

[62] World Health Organization. *Family of International Classifications (FIC)*. URL: `https://www.who.int/classifications/en` (visited on 2020-07-26).

[63] Ontology Engineering Group. *Linked Open Vocabularies (LOV)*. URL: `https://lov.linkeddata.es/dataset/lov/` (visited on 2020-07-26).

[64] C. Atkinson and K. Kiko. *A Detailed Comparison of UML and OWL*. Technical Report. Department for Mathematics and Computer Science, University of Mannheim, June 2005. URL: `https://madoc.bib.uni-mannheim.de/1898/`.

[65] B. Glimm et al. "HermiT: An OWL 2 Reasoner." In: *Journal of Automated Reasoning* 53.3 (Oct. 2014), pp. 245–269. ISSN: 1573-0670. DOI: `10.1007/s10817-014-9305-1`.

[66] E. Sirin et al. "Pellet: A practical OWL-DL reasoner." In: *Journal of Web Semantics* 5.2 (June 2007), pp. 51–53. ISSN: 1570-8268. DOI: `10.1016/j.websem.2007.03.004`.

[67] P. Hitzler et al. *Semantic Web: Grundlagen*. eXamen.press. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008. ISBN: 9783540339939. DOI: `10.1007/978-3-540-33994-6`.

[68] E. A. Locke and G. P. Latham. *A Theory of Goal Setting & Task Performance*. Englewood Cliffs, N.J.: Prentice Hall, Jan. 1990. ISBN: 978-0139131387.

[69] F. Gleich. *TISAX Participant Handbook*. Frankfurt am Main, Germany: ENX Association, 2020. URL: `https://portal.enx.com/tphen.pdf` (visited on 2020-11-10).

[70] K. Boeckl et al. *NISTIR 8228: Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks*. Standard. National Institute of Standards and Technology (NIST), June 2019. DOI: `10.6028/NIST.IR.8228`.

[71] Industrial Internet Consortium. *The Industrial Internet of Things Volume G1: Reference Architecture*. Standard. IIC, Jan. 2017.

[72] Deutsches Institut für Normung e. V. *DIN SPEC 91345, Reference Architecture Model Industrie 4.0 (RAMI4.0).* Standard. Berlin, Germany: DIN, Apr. 2016.

[73] S.-W. Lin et al. *Architecture Alignment and Interoperability - An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper*. Whitepaper IIC:WHT:IN3:V1.0:PB:20171205. Industrial Internet Consortium, Dec. 2017. URL: https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf (visited on 2020-07-26).

[74] International Organization for Standardization. *ISO 28000:2007 Specification for security management systems for the supply chain*. Standard. Geneva, CH: ISO, Sept. 2007.

[75] B. Hellingrath and M. ten Hompel. "IT & Forecasting in Industrie und Handel." In: *Effizienz - Verantwortung - Erfolg. 24. Deutscher Logistik-Kongress* (Oct. 2007).

[76] D. Gritzalis et al. "Exiting the Risk Assessment Maze: A Meta-Survey." In: *ACM Comput. Surv.* 51.1 (Apr. 2018), 11:1–11:30. ISSN: 0360-0300. DOI: 10.1145/3145905.

[77] K. Waldron. *Resources for Measuring Cybersecurity. A Partial Annotated Bibliography*. Technical Report. Washington, D.C., USA: R Street Institute, Oct. 2019. DOI: 10.6028/NIST.CSWP.04162018.

[78] B. Fabian et al. "A comparison of security requirements engineering methods." In: *Requirements Engineering* 15.1 (Mar. 2010), pp. 7–40. DOI: 10.1007/s00766-009-0092-x.

[79] X. Ou, S. Govindavajhala, and A. W. Appel. "MulVAL: A Logic-based Network Security Analyzer." In: *Proceedings of the 14th conference on USENIX Security Symposium*. SSYM'05. Baltimore, MD, USA: USENIX Association, July 2005. URL: http://dl.acm.org/citation.cfm?id=1251398.1251406 (visited on 2020-07-26).

[80] R. Lippmann et al. "Validating and Restoring Defense in Depth Using Attack Graphs." In: *MILCOM 2006 - 2006 IEEE Military Communications conference*.

Piscataway, NJ, USA: IEEE, Oct. 2006, pp. 1–10. ISBN: 1-4244-0617-X. DOI: 10.1109/MILCOM.2006.302434.

[81] S. Noel et al. "Advances in Topological Vulnerability Analysis." In: *2009 Cybersecurity Applications Technology Conference for Homeland Security*. Piscataway, NJ, USA: IEEE, Mar. 2009, pp. 124–129. ISBN: 978-0-7695-3568-5. DOI: 10.1109/CATCH.2009.19.

[82] S. Jajodia et al. "Cauldron mission-centric cyber situational awareness with defense in depth." In: *2011 - MILCOM 2011 Military Communications Conference*. MILCOM '11. Piscataway, NJ, USA: IEEE, Nov. 2011, pp. 1339–1344. DOI: 10.1109/MILCOM.2011.6127490.

[83] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. "DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees." In: *Computer science review* 13 (Nov. 2014), pp. 1–38. DOI: 10.1016/j.cosrev.2014.07.001.

[84] C. Bock et al. *Unified Modeling Language*. Specification. Needham, MA, USA: Object Management Group (OMG), Dec. 2017.

[85] T. Lodderstedt, D. Basin, and J. Doser. "SecureUML: A UML-Based Modeling Language for Model-Driven Security." In: *UML 2002 — The Unified Modeling Language*. Ed. by J.-M. Jézéquel, H. Hussmann, and S. Cook. Berlin, Heidelberg: Springer Berlin Heidelberg, Sept. 2002, pp. 426–441. ISBN: 978-3-540-45800-5. DOI: 10.1007/3-540-45800-X_33.

[86] J. Jürjens. "UMLsec: Extending UML for Secure Systems Development." In: *UML 2002 — The Unified Modeling Language*. Ed. by J.-M. Jézéquel, H. Hussmann, and S. Cook. Berlin, Heidelberg: Springer Berlin Heidelberg, Sept. 2002, pp. 412–425. ISBN: 978-3-540-45800-5. DOI: 10.1007/3-540-45800-X_32.

[87] Y. Ji et al. "A Logic-based Approach to Network Security Risk Assessment." In: *International Colloquium on Computing, Communication, Control, and Management*. ISECS '09. Sanya, China: IEEE, Sept. 2009. ISBN: 978-1-4244-4247-8. DOI: 10.1109/CCCM.2009.5267887.

[88] R. Zakeri et al. "Using Description Logics for Network Vulnerability Analysis." In: *Fifth International Conference on Networking and the International Conference on Systems*. ICNICONSMCL'06). Mauritius: IEEE, Apr. 2006, p. 78. ISBN: 0-7695-2552-0. DOI: 10.1109/ICNICONSMCL.2006.222.

[89] A. Kim, J. Luo, and M. Kang. "Security Ontology for Annotating Resources." In: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*. Ed. by R. Meersman and Z. Tari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1483–1499. ISBN: 978-3-540-32120-0. DOI: 10.1007/11575801_34.

[90] S. Fenz and A. Ekelhart. *Formalizing information security knowledge*. New York, NY, USA: Association for Computing Machinery, 2009. ISBN: 978-1-60558-394-5. DOI: 10.1145/1533057.1533084.

[91] A. Souag et al. "A Security Ontology for Security Requirements Elicitation." In: *Engineering Secure Software and Systems*. Ed. by F. Piessens, J. Caballero, and N. Bielova. ESSoS 2015. Cham, Germany: Springer International Publishing, Mar. 2015, pp. 157–177. ISBN: 978-3-319-15618-7. DOI: 10.1007/978-3-319-15618-7_13.

[92] M. J. O'Connor and A. K. Das. "SQWRL: A Query Language for OWL." In: *Proceedings of the 6th International Conference on OWL: Experiences and Directions*. Vol. 529. OWLED'09. Chantilly, VA, USA: CEUR-WS.org, Oct. 2009, pp. 208–215. URL: http://dl.acm.org/citation.cfm?id=2890046.2890072 (visited on 2020-07-26).

[93] F. Patzer et al. "Towards Computer-aided Security Life Cycle Management for Critical Systems." In: *13th International Conference on Critical Information Infrastructures Security*. CRITIS '18. Cham, Germany: Springer, Dec. 2018, pp. 45–56. DOI: 10.1007/978-3-030-05849-4_4.

[94] S. N. Foley and W. M. Fitzgerald. "Management of security policy configuration using a Semantic Threat Graph approach." In: *Journal of Computer Security* 19.3 (May 2011), pp. 567–605. DOI: 10.3233/JCS-2011-0421.

[95] L. Kohnfelder and P. Garg. *The threats to our products*. Apr. 1999. URL: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx (visited on 2020-07-26).

[96] A. Shostack. *Threat Modeling: Designing for Security*. Indianapolis, IN, USA: John Wiley and Sons, Feb. 2014. ISBN: 9781118809990.

[97] M. S. Lund, B. Solhaug, and K. Stølen. *Model-Driven Risk Analysis: the CORAS approach*. Springer Science & Business Media, Oct. 2010. ISBN: 978-3642123238. DOI: 10.1007/978-3-642-12323-8.

[98] T. Sommestad, M. Ekstedt, and H. Holm. “The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures.” In: *IEEE Systems Journal* 7.3 (Dec. 2013), pp. 363–373. ISSN: 1932-8184. DOI: 10.1109/JSYST.2012.2221853.

[99] H. Holm et al. “P^{2}CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language.” In: *IEEE Transactions on Dependable and Secure Computing* 12.6 (Nov. 2015), pp. 626–639. ISSN: 1545-5971. DOI: 10.1109/TDSC.2014.2382574.

[100] J. Freund and J. Jones. *Measuring and Managing Information Risk: A FAIR approach*. Oxford, UK: Butterworth-Heinemann, 2015. ISBN: 9780124202313.

[101] D. Angermeier, A. Nieding, and J. Eichler. “Supporting Risk Assessment with the Systematic Identification, Merging, and Validation of Security Goals.” In: *Risk Assessment and Risk-Driven Quality Assurance*. Ed. by J. Großmann, M. Felderer, and F. Seehusen. Cham: Springer International Publishing, Apr. 2017, pp. 82–95. ISBN: 978-3-319-57858-3.

[102] D. Angermeier, A. Nieding, and J. Eichler. “Systematic Identification of Security Goals and Threats in Risk Assessment.” In: *Softwaretechnik-Trends* 36.3 (2016).

[103] M. Krisper et al. “RISKEE: A Risk-Tree Based Method for Assessing Risk in Cyber Security.” In: *Systems, Software and Services Process Improvement - 26th European Conference, EuroSPI 2019, Edinburgh, UK, September 18-20, 2019, Proceedings*. 2019, pp. 45–56. DOI: 10.1007/978-3-030-28005-5_4.

[104] D. Firesmith. “Specifying reusable security requirements.” In: *Journal of Object Technology* 3.1 (Jan. 2004), pp. 61–75.

[105] S. J. Greenspan, A. Borgida, and J. Mylopoulos. "A Requirements Modeling Language and Its Logic." In: *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*. Ed. by M. L. Brodie and J. Mylopoulos. New York, NY, USA: Springer New York, 1986, pp. 471–502. ISBN: 978-1-4612-4980-1. DOI: `10.1007/978-1-4612-4980-1_37`.

[106] S. Greenspan, J. Mylopoulos, and A. Borgida. "On formal requirements modeling languages: RML revisited." In: *Proceedings of 16th International Conference on Software Engineering*. ICSE '94. Piscataway, NJ, USA: IEEE, May 1994, pp. 135–147. DOI: `10.1109/ICSE.1994.296773`.

[107] F. Patzer et al. "Towards the modelling of complex communication networks in AutomationML." In: *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Piscataway, NJ, USA: IEEE, Sept. 2017. DOI: `10.1109/ETFA.2017.8247571`.

[108] C. Ebert and M. Jastram. "ReqIF: Seamless Requirements Interchange Format between Business Partners." In: *IEEE Software* 29.5 (Sept. 2012), pp. 82–87. DOI: `10.1109/MS.2012.121`.

[109] B. Zhu and S. Sastry. "SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy." In: *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*. Vol. 11. SCS '10. 2010, p. 7.

[110] R. Mitchell and I.-R. Chen. "A Survey of Intrusion Detection Techniques for Cyber-physical Systems." In: *ACM Computing Surveys* 46.4 (Mar. 2014), 55:1–55:29. ISSN: 03600300. DOI: `10.1145/2542049`.

[111] D. Hadžiosmanović et al. "Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes." In: *Proceedings of the 30th Annual Computer Security Applications Conference*. ACSAC '14. New Orleans, Louisiana, USA: Association for Computing Machinery, Dec. 2014, pp. 126–135. ISBN: 978-1-4503-3005-3. DOI: `10.1145/2664243.2664277`.

[112] M. Caselli, E. Zambon, and F. Kargl. "Sequence-aware Intrusion Detection in Industrial Control Systems." In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. CPSS '15. Singapore, Republic of Singapore:

Association for Computing Machinery, Apr. 2015, pp. 13–24. ISBN: 978-1-4503-3448-8. DOI: 10.1145/2732198.2732200.

[113] P. Haller and B. Genge. “Using sensitivity analysis and cross-association for the design of intrusion detection systems in industrial cyber-physical systems.” In: *IEEE Access* 5 (May 2017), pp. 9336–9347. DOI: 10.1109/ACCESS.2017.2703906.

[114] F. Pasqualetti, F. Dörfler, and F. Bullo. “Attack Detection and Identification in Cyber-Physical Systems.” In: *IEEE Transactions on Automatic Control* 58.11 (Nov. 2013), pp. 2715–2729. DOI: 10.1109/TAC.2013.2266831.

[115] S. Potluri, C. Diedrich, and G. K. R. Sangala. “Identifying False Data Injection Attacks in Industrial Control Systems using Artificial Neural Networks.” In: *22nd IEEE International Conference on Emerging Technologies and Factory Automation*. ETFA ’17. Piscataway, NJ, USA: IEEE, Jan. 2017, pp. 1–8. DOI: 10.1109/ETFA.2017.8247663.

[116] M. Nicolett and K. M. Kavanagh. *Magic Quadrant for Security Information and Event Management*. Gartner RAS Core Research Note G00212454. Gartner Research, 2011.

[117] M. Mantere, M. Sailio, and S. Noponen. “Network Traffic Features for Anomaly Detection in Specific Industrial Control System Network.” In: *Future Internet* 5.4 (Dec. 2013), pp. 460–473. DOI: 10.3390/fi5040460.

[118] E. Pleijsier. “Towards anomaly detection in SCADA networks using connection patterns.” In: *18th Twente Student Conference on IT*. Enschede,The Netherlands, 2013, pp. 1–6.

[119] T. Slot and F. Kargl. “Detection of APT Malware through External and Internal Network Traffic Correlation.” PhD thesis. Master’s thesis, University of Twente, Mar. 2015.

[120] R. Udd et al. “Exploiting Bro for Intrusion Detection in a SCADA System.” In: *Proceedings of the 2Nd ACM International Workshop on Cyber-Physical System Security*. CPSS ’16. Xi’an, China: Association for Computing Machinery, May 2016, pp. 44–51. ISBN: 978-1-4503-4288-9. DOI: 10.1145/2899015.2899028.

[121] Y. Yang et al. “Stateful intrusion detection for IEC 60870-5-104 SCADA security.” In: *2014 IEEE Power & Energy Society General Meeting*. Piscataway, NJ, USA: IEEE, July 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939218.

[122] L. A. Maglaras and J. Jiang. “Intrusion detection in SCADA systems using machine learning techniques.” In: *2014 Science and Information Conference (SAI)*. Aug. 2014, pp. 626–631. DOI: 10.1109/SAI.2014.6918252.

[123] K. Veeramachaneni et al. “AI^2: Training a Big Data Machine to Defend.” In: *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. Piscataway, NJ, USA: IEEE, Apr. 2016, pp. 49–54. ISBN: 978-1-5090-2403-2. DOI: 10.1109/BigDataSecurity-HPSC-IDS.2016.79.

[124] R. Lippmann et al. “The 1999 DARPA Off-line Intrusion Detection Evaluation.” In: *Computer networks* 34.4 (Oct. 2000), pp. 579–595. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(00)00139-0.

[125] A. Lemay and J. M. Fernandez. “Providing SCADA Network Data Sets for Intrusion Detection Research.” In: *9th Workshop on Cyber Security Experimentation and Test*. CSET ’16. Austin, TX, USA: USENIX Association, Aug. 2016. URL: https://www.usenix.org/conference/cset16/workshop-program/presentation/lemay (visited on 2020-07-26).

[126] NETRESEC. *Capture files from 4SICS Geek Lounge*. URL: https://www.netresec.com/?page=PCAP4SICS (visited on 2020-07-26).

[127] A. Giehl and N. Wiedermann. “Security Verification of Third Party Design Files in Manufacturing.” In: *Proceedings of the 2018 10th International Conference on Computer and Automation Engineering*. ICCAE ’2018. Brisbane, Australia: Association for Computing Machinery, Feb. 2018, pp. 166–173. ISBN: 978-1-4503-6410-2. DOI: 10.1145/3192975.3192984.

[128] P. T. Vlacheas et al. *Ontology and taxonomies of resilience*. Report. Attiki, Greece: European Union Agency for Cybersecurity (ENISA), Dec. 2011, p. 59.

URL: https://www.enisa.europa.eu/publications/ontology_taxonomies (visited on 2020-07-26).

[129] D. A. Mundie et al. “An Incident Management Ontology.” In: *STIDS*. Vol. 1304. Jan. 2014, pp. 62–71.

[130] H. Bronk, M. Thorbruegge, and M. Hakkaja. *A STEP-BY-STEP APPROACH ON HOW TO SET UP A CSIRT*. Deliverable. Attiki, Greece: European Union Agency for Cybersecurity (ENISA), Dec. 2006. URL: https://www.enisa.europa.eu/publications/csirt-setting-up-guide (visited on 2020-07-26).

[131] ENISA. *Detect, SHARE, Protect: Solutions for Improving Threat Data Exchange among CERTs*. Report. Attiki, Greece: European Union Agency for Cybersecurity (ENISA), Nov. 2013. URL: https://www.enisa.europa.eu/activities/cert/support/information-sharing/detect-share-protect-solutions-for-improving-threat-data-exchange-among-certs (visited on 2020-07-26).

[132] ENISA. *Analysis of ICS-SCADA Cyber Security Maturity Levels in Critical Sectors*. Report. Attiki, Greece: European Union Agency for Cybersecurity (ENISA), Dec. 2015. DOI: 10.2824/835661. URL: https://www.enisa.europa.eu/publications/maturity-levels (visited on 2020-07-26).

[133] European Commission. *Proposal for a DIRECTIVE OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL concerning measures to ensure a high common level of network and information security across the Union*. Report. EC, Feb. 2013.

[134] M. Sharif et al. “Automatic Reverse Engineering of Malware Emulators.” In: *30th IEEE Symposium on Security and Privacy*. S&P ’09. Piscataway, NJ, USA: IEEE, May 2009, pp. 94–109. DOI: 10.1109/SP.2009.27.

[135] U. Bayer et al. “Scalable, Behavior-Based Malware Clustering.” In: *Proceedings of the Network and Distributed System Security Symposium*. Vol. 9. NDSS ’09. San Diego, CA, USA: The Internet Society, Feb. 2009, pp. 8–11. URL: https://www.ndss-symposium.org/ndss2009/scalable-behavior-based-malware-clustering/.

[136] OASIS Open. *STIX™ Version 2.1*. Committee Specification. OASIS Cyber Threat Intelligence (CTI) TC, Mar. 2020. URL: `https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.pdf` (visited on 2020-07-26).

[137] L. A. F. Martimiano and E. d. S. Moreira. "The Evaluation Process of a Computer Security Incident Ontology." In: *Proceedings of the Workshop on 2nd Workshop on Ontologies and their Applications*. WONTO '06. Ribeirao Preto, SP, Brazil: CEUR-WS.org, Oct. 2006. URL: `http://ceur-ws.org/Vol-199/wonto-06.pdf` (visited on 2020-07-26).

[138] H. Kaufmann et al. "A structural design for a pan-European early warning system for critical infrastructures." In: *e & i Elektrotechnik und Informationstechnik* 132 (Feb. 2015). DOI: `10.1007/s00502-015-0286-5`.

[139] M. P. Barrett. *Framework for Improving Critical Infrastructure Cybersecurity*. Tech. rep. National Institute of Standards and Technology (NIST), Apr. 2018. DOI: `10.6028/NIST.CSWP.04162018`.

[140] J. Bruinenberg et al. *CEN-CENELEC-ETSI Smart Grid Coordination Group Smart Grid Reference Architecture*. Technical Report. CEN, CENELEC, ETSI, Nov. 2012. URL: `https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf` (visited on 2020-07-26).

[141] G. E. P. Box. "Robustness in the Strategy of Scientific Model Building." In: *Robustness in Statistics*. Ed. by R. L. Launer and G. N. Wilkinson. Academic Press, 1979, pp. 201–236. ISBN: 978-0-12-438150-6. DOI: `10.1016/B978-0-12-438150-6.50018-2`.

[142] International Electrotechnical Commission. *IEC 61508-1:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements*. Standard. Geneva, CH: IEC, Apr. 2010.

[143] International Organization for Standardization. *Road vehicles — Functional safety — Part 1: Vocabulary*. Standard. Geneva, CH: ISO, Dec. 2018.

[144] P. Schneider and K. Böttinger. "High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks." In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. CPS-SPC '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 1–12. ISBN: 978-1-4503-5992-4. DOI: 10.1145/3264888.3264890.

[145] FreeOpcUa. *FreeOpcUa: Open Source C++ and Python OPC-UA Server and Client Libraries and Tools*. 2019. URL: http://freeopcua.github.io/ (visited on 2020-07-26).

[146] General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China. *Technical specifications of remote service and management system for electric vehicles - Part 1: General Principle*. Standard. Peking, CN: AQSIQ, Nov. 2016.

[147] R. Langner. "Stuxnet: Dissecting a Cyberwarfare Weapon." In: *IEEE Security & Privacy* 9.3 (May 2011), pp. 49–51. DOI: 10.1109/MSP.2011.67.

[148] SophosLabs Research Team and others. "Emotet exposed: looking inside highly destructive malware." In: *Network Security* 2019.6 (June 2019), pp. 6–11. ISSN: 1353-4858. DOI: 10.1016/S1353-4858(19)30071-6.

[149] N. Nell. *The Impact of Dragonfly Malware on Industrial Control Systems*. STI Graduate Student Research. SANS Institute, Jan. 2016.

[150] P. Cichonski et al. *SP 800-61 Rev. 2: Computer Security Incident Handling Guide*. Standard. National Institute of Standards and Technology (NIST), Aug. 2012. DOI: 10.6028/NIST.SP.800-61r2.

[151] G. Rizzo et al. "World's Largest Electric Vehicle Operation in Mines." In: *ATZoffhighway worldwide* 11.4 (Oct. 2018), pp. 12–18. ISSN: 2366-1097. DOI: 10.1007/s41321-018-0041-z.

[152] B. Varocky et al. *Benchmarking of Regenerative Braking for a Fully Electric Car*. Internship Report. TNO Automotive, Helmond & Technische Universiteit Eindhoven (TU/e), Jan. 2011. URL: http://www.mate.tue.nl/mate/pdfs/12673.pdf (visited on 2020-07-26).

[153] U. Blumenthal and P. Goel. *Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)*. Tech. rep. Internet Engineering Task Force (IETF), Jan. 2007. 5 pp. DOI: `10.17487/RFC4785`.

[154] M. Muckin and S. C. Fitch. *A Threat-Driven Approach to Cyber Security: Methodologies , Practices and Tools to Enable a Functionally Integrated Cyber Security Organization*. Whitepaper. Lockheed Martin Corporation, 2014. URL: `https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Threat-Driven-Approach.pdf` (visited on 2020-07-26).

List of Abbreviations

ACL	Access-Control List
API	Application Programming Interface
AT	Attack Tree
BOK	Body of Knowledge
CI	Critical Infrastructure
CIL	Cybersecurity Impact Level
CIL_{Aut}	Authenticity Impact Level
CIL_{Ava}	Availability Impact Level
CIL_{Conf}	Confidentiality Impact Level
CIL_{Int}	Integrity Impact Level
CIL_{NR}	Non-Repudiation Impact Level
COTS	Commercial-off-the-Shelf
CPE	Common Platform Enumeration
CPS	Cyber-Physical System
CSIRT	Computer Security Incident Response Team
CWE	Common Weakness Enumeration
DSML	Domain Specific Modeling Language
E-SOC	Security Operations Center at European Level
ECU	Electronic Control Unit
ERP	Enterprise Resource Planning
ESP	Electronic Stability Control
EV	Electronic Vehicle

FFC	Fine-grained Feature Classification
HSM	Hardware Security Module
ICS	Industrial Control System
ICT	Information and Communications Technology
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
INDS	Industrial Data Space
IoC	Indicator of Compromise
IoT	Internet of Things
IPS	Intrusion Prevention System
IUNO	German National Reference Project for IT Security in Industry 4.0
KM	Key Management
LKG	Local Knowledge Graph
M2M	Machine-to-Machine
MoRA	Modular Risk Assessment
N-SOC	Security Operations Center at National Level
NIDS	Network Intrusion Detection Systems
O-SOC	Security Operations Center at Organizational Level
OPC-UA	OLE for Process Control – Unified Architecture
OT	Operational Technology
OWL	Web Ontology Language
PAP	Provided Attack Potential

PLC	Programmable Logic Controller
RAP	Required Attack Potential
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
ReqIF	Requirements Interchange Format
SCADA	Supervisory Control and Data Acquisition
SI	Secure Implementation
SIEM	Security Information and Event Management
SNC	Secure Network Configuration
SOC	Security Operations Center
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SRA	Security Risk Assessment
SSML	Security-specific Modeling Language
SuE	System under Evaluation
TAP	Network Wiretap Device
UML	Unified Modeling Language
URI	Uniform Resource Identifier
ViWi	Volkswagen infotainment Web interface protocol
VPN	Virtual Private Network

List of Figures

List of Tables