# Deriving Impact-driven Security Requirements and Monitoring Measures for Industrial IoT

Gerhard Hansch
gerhard.hansch@aisec.fraunhofer.de
Fraunhofer AISEC
Garching, Germany

Peter Schneider
peter.schneider@aisec.fraunhofer.de
Fraunhofer AISEC
Garching, Germany

Gerd Stefan Brost
gerd.brost@aisec.fraunhofer.de
Fraunhofer AISEC
Garching, Germany

## ABSTRACT

The emerging Industrial Internet of Things (IIoT) is characterized by heterogeneous systems, loose topologies, cross-company data flows, changing entities, and high cybersecurity requirements. This development makes a sound security architecture an even more pressing matter than before. The design of a valid security architecture should always reflect the protection needs, enable the derivation of security requirements, and ways to validate their effectiveness. While access management at the application layer is well established, securing the underlying network layers of an increasing number of communication links remains an open question. Currently, adapting to the dynamics of rapid technology changes requires reiterating time- and resource-intensive threat and risk analyses. Therefore, we introduce and apply a lightweight, graph-based process to create easy-to-build and machine-readable models of the reviewed IIoT systems, highlighting the assets they contain. Such a model allows us to derive abstract security requirements in a semi-automatic way. We use these requirements to propose appropriate protection and advanced monitoring measures as well as methods to validate their effective implementation. The catalog provided in this paper represents a security toolbox for these two security layers, tailored for the IIoT domain. Finally, it allows for deriving rules for current anomaly detection solutions. Thus, we support the often-laborious definition and prioritization of monitoring rules by an impact-based automated approach. By connecting the catalog with the lightweight impact analysis, we provide a framework that dynamically derives recommendations and requirements from a variety of monitoring measures and techniques. Thereby we provide a general methodology that helps operators to strengthen the overall security of their IIoT systems.

## CCS CONCEPTS

• **Security and privacy → Security requirements**; *Intrusion detection systems*; *Information flow control*; • **Software and its engineering → Requirements analysis**; Risk management; • **Networks →** *Network security*.

## KEYWORDS

Cyber-Physical System Security, IIoT, Modeling, Network Monitoring, Protection Measures, Requirement Prioritization, Security Measure Monitoring, Security Objective Relevance

*CPSS '19, July 8, 2019, Auckland, New Zealand*


**ACM Reference Format:**
Gerhard Hansch, Peter Schneider, and Gerd Stefan Brost. 2019. Deriving Impact-driven Security Requirements and Monitoring Measures for Industrial IoT. In *5th ACM Cyber-Physical System Security Workshop (CPSS '19), July 8, 2019, Auckland, New Zealand*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3327961.3329528

## 1 INTRODUCTION

The Fourth Industrial Revolution means a fundamental shift from isolated production to demand-specific reconfigurations of data flows and production lines. To minimize losses due to cybersecurity incidents, emerging challenges and risks must be identified and managed. While many findings from the discussion about cybersecurity for consumer Internet of Things (IoT) devices [34] should be transferable to the industry, different life cycles, infrastructures, and values of Industrial Internet of Things (IIoT) require additional planning and efforts.

Industrial production line devices are often expensive, long-term investments requiring decades of support and maintenance. Many of these systems already support software- and hardware-updates, but lack appropriate protection for flexible data flows, which are up to now reserved for server IT. This includes service-oriented architectures (SoA), software-defined networking (SDN), and the spontaneous addition or removal of sensors, actuators, and data applications. The resulting rerouting can suddenly change the current threat situation due to shifting dependencies or exposures. This raises the question of how the lower network layers and routes used by more and more communication partners can be secured.

This development demands an automatism that can quickly assess the current situation and provide appropriate security strategies for each new configuration, as well as adaptive security controls. While an experienced operator 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. Our approach proposes such measures based on an IIoT system model to support the operator in making decisions.

Conventional networked devices serve a specific purpose and have a limited number of communication partners and data flows. In a dynamic IIoT-network with $n$ nodes, the number of communication links and flows can exhibit square growth with the addition of new sensors, data applications, actuators and endpoints. As each of them is able to communicate with all others, one new network node may lead to $n$ new connections. Due to the comprehensive networking of IIoT-devices in a dynamic industrial environment,

risks can no longer be individually identified and assessed. Segmentation and gateways in practical operation reduce this growth from exponential to a polynomial level. An automated approach is nonetheless inevitable for responsible operation in cybersecurity-relevant infrastructures.

A proven method to understand the security needs of complex IT systems is a model-based impact analysis. On the basis of such an analysis, requirements for protective measures can be derived and prioritized, as can their evaluation and continuous monitoring.

To create easy-to-build models of the IIoT systems under evaluation, we introduce and use a domain specific modeling language (DSML), based on the Web Ontology Language (OWL) to describe the systems and the valuable assets. Given a model in this language, we perform queries to identify the critical elements that require protection or monitoring.

We derive appropriate non-invasive measures for these, based on a catalog of monitoring- and detection-based mechanisms relying on already available information and the system model. Hence, we assist in the process of rule definition for these systems using an interactive approach.

Thus, we make the following contributions in this paper:

- An interactive cybersecurity impact assessment method (cf. Chapter 3) to identify valuable assets and their protection needs.
- A method to derive security requirements and protection measures from the outcome of the assessment (cf. Chapter 4).
- A catalog of detection and monitoring techniques to determine the current security status (cf. Chapter 5).

These contributions and how they work together are demonstrated in a running example, based on a real-world use case.

## 2 RELATED WORK

With the so called Fourth Industrial Revolution, industries started to connect production machines to internal and remote networks [21]. This enhanced connectivity from remote services down through sensors and actuators in shop floors enables new business models. By that, these machines form cyber-physical systems (CPS) building an IIoT. The Industrial Internet of Things Reference Architecture [17] and the Reference Architecture Model Industrie 4.0 [10] are two reference architectures, each presenting numerous concepts and methods from different perspectives. The common goal of both architectures is to support their users in creating implementation concepts [25] by reaching a common understanding. While [17] concentrates on vertical cross-domain and interoperability, [10] focuses on the horizontal value chain of manufacturing. The use case analyzed in Chapter 3.3 originates from the IIoT-specific architecture described in [6].

Bridging the gap between the virtual and physical worlds, CPS introduce new attack vectors [3]. Previous attacks such as Stuxnet and Duqu showed damage potentials ranging from theft of intellectual property to physical damage [8]. While operators of these facilities deploy security measures from the business IT in [4, 8, 23], the authors give insights on recent attacks, common protection goals, and implementation strategies for IIoT requiring a much more in-depth risk analysis and security measure implementation [34]. A blueprint for a data space architecture that allows for exchange of data from different domains (IoT, medical, chemical) is presented in [28].

The quality of a security architecture depends on the quality of the security requirements. An overview of approaches and techniques for security requirements engineering is provided by [11]. The authors review and compare in detail eighteen methods (including CORAS, Misuse Cases, Secure Tropos and UMLSec).

A fundamental part of a security management strategy [19] is the identification of information security risks, either in operation [20, 22] or product development [9, 20]. Unfortunately, the identification and assessment of valuable assets is a hard challenge for several methods. [2, 42] address this topic with quantified risk analysis methods, focusing on the automotive sector. [2] creates tuples for the violation of each information security goal for every function, data and component, and a predefined, discrete matrix of different impact factors. Depending on the assumed impact, the individual protection need is determined. In practice, the manual assignment of discrete values from a predefined list to individual assets proved to be a real challenge and error-prone task.

A valid network model in a knowledge representation language, combined with semantic reasoning opens a multitude of different application possibilities. It can effectively be used as part of an interactive systems engineering process [31], to generate network rules [12], or to find known vulnerabilities [29, 41]. Wolf et al. [41] introduced an OWL ontology and tailored formal method to selectively refine models as part of a vulnerability analysis. While the application purpose and modeling granularity are different, several modeling concepts (such as the classes and relationships) are equivalent to those used in this paper, providing a link for model reuse.

To limit access to resources, traditionally, high level access control mechanisms are used. Usage control is a suitable measure to limit how data may be used after initial access has been granted and the data has left the control domain of the initial owner. A label based, information flow control approach for our running example was presented in [37]. In contrast to their work, we focus on lower application layers to monitor and control data flows.

The monitoring for anomalous behavior is referred to as anomaly detection. Anomaly detection for IIoT needs to incorporate different data sources and types. Hence, a lot of research focused on anomaly detection methods for specific data types [7, 14, 15, 30, 32]. For a holistic monitoring and detection system, however, we need to monitor all data types. To aggregate these differing data types, industries developed security information and event management systems (SIEM). An overview of existing systems is given in [27].

## 3 MODEL-BASED IMPACT ANALYSIS

To identify assets and determine their need for protection, we perform a qualitative impact analysis. Dependencies and potential negative impacts are analyzed by creating and assessing a semantic model of the system under evaluation.

The DSML applied for the creation of such models is defined and visualized in Chapter 3.1. A procedure to determine, evaluate and propagate security objectives, leading to individual security relevance ratings is described in Chapter 3.2. The process of modeling

and analyzing an exemplary use case is demonstrated in Chapter 3.3.

## 3.1 Domain Specific Modeling Language

Given the dynamic communication behavior in the IIoT, assets and dependencies are subject to spontaneous change. Thus, the model of an IIoT system itself is subject to frequent change and must be easily maintainable. The creation of a useful model requires a suitable modeling language. While expressive modeling languages offer more detailed models, a simple language is sufficient for our analysis. The lightweight DSML we introduce for this task is based on a set of basic classes (functions, components, connections and data) and corresponding relations as shown in Figure 1. If necessary, a hierarchical refinement as described by [41] is possible.

A *Function* is defined as 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 are cyber-physical functions, which consume and produce data within the model, but also operate outside of it. *Components* are (physical or virtual) instances that host functions or store data. They may consist of several sub-components [41]. The transfer of data between functions takes place via *Connections*. *Data* is any type of information that can be consumed or produced by a function, stored at a component or transmitted via a connection. These basic *Classes* alongside their *Relations* are shown in Figure 1.

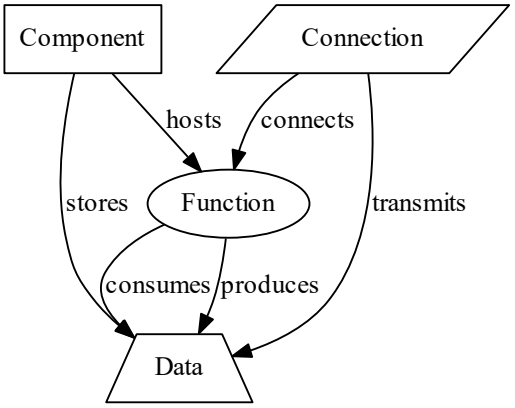


**Figure 1: Meta-model of the modeling language.**

The notation for describing the infrastructure and data flows can be derived from the meta-model. Clustering the static and dynamic elements during modeling is recommended as it improves maintainability. A more detailed categorization of the instances is necessary to select appropriate actions. In order to avoid unnecessary effort and to favor a simple model, such additional information is augmented later as required.

## 3.2 Security Objective Relevance Rating

After the creation of a combined infrastructure and data flow model, the individual severity of missing a security objective (i.e. confidentiality, integrity, availability) can be determined. For this purpose, the security objectives of each instance (i.e. function, component, connection, and data) have security relevance levels (SRL) assigned to them, which lead to a number of weighted assets.

With the following method we calculate the SRL of all instances of the models iteratively. Note that only a small set of initial data $D_0$ (for confidentiality) and sink-function $F_0$ (for integrity and availability) must be rated manually in order to propagate the SRL through the remaining model. To cope with special cases, relevance ratings can still be adjusted manually along the propagation paths.

The output of this process is a list of assets with their individual protection needs. In contrast to static networks where it was enough to assess the infrastructure once and then inherit security goals from the data, this process should be triggered on each update of the model.

*Confidentiality Relevance Level:* To evaluate the confidentiality-specific SRL of all instances, we start with an initial set of manually rated data $D_0$ from the source functions. The main idea here is that the confidentiality-specific SRL of $D_0$ propagate down through the remaining model and we can recursively evaluate the SRL of all instances iteratively. First, we evaluate the confidentiality SRL of a function to be the maximum confidentiality SRL of the data that are consumed by this function and if applicable an intrinsic value (e.g., a valuable sophisticated algorithm). Second, we evaluate the SRL of a connection to be the maximum SRL of data that are transferred through this connection. Third, we evaluate the SRL of a component to be the maximum SRL of functions that are hosted by this component. Fourth, we evaluate the SRL of previously unrated data to be the SRL of connections they are transmitted through, functions they are produced by, or components they are stored at, respectively.

*Integrity and Availability Relevance Level:* The process to evaluate the integrity and availability relevance of all instances propagates bottom-up, unlike the top-down approach of the confidentiality-related process. We start with an initial set of manually rated sink-functions $F_0$. The main idea here is that the integrity-, respectively availability-specific SRL of $F_0$ propagates up through the whole model and we can recursively evaluate the SRL of all instances iteratively. First, we evaluate the SRL of a data consumed by $F_0$ to be the maximum SRL of the function that consumes this data. Second, we evaluate the SRL of a connection to be the maximum SRL of data that are transferred through this connection. Third, we evaluate the SRL of a component to be the maximum SRL of functions that are hosted by this component. Fourth, we evaluate the SRL of previously unrated functions to be the level of data they produce. The integrity SRL can individually be adjusted if the function has external effects or supports other security objectives, as a filter function does for data.

## 3.3 Use Case Example

To demonstrate our framework, we use an example, based on a real-word use case from the Industrial Dataspace (IDS) [28]. The IDS aims at creating a data marketplace which enables participants to share and consume data and services. *Connectors* act as edge gateways and connect different trust domains. They host data services which allow for data collection, enrichment and filtering. The

analysis of this use case, which is visualized in Figure 2, is the baseline for the propagation example show in in Table 1.

Sensors (**C_S**ensor **X**) are components on their own. Via their functions (**F_S**ensor **X**), they generate sensor data (**D_S**ensor **X**), e.g., production parameters, and transmit them via connections (**N_S**ensor **X**) to a service (**F_P**roduction **S**ervice), hosted at a Connector (**C_C**onnector **A**). The aggregated data (**D_A**ggregated **D**ata) is forwarded via a connection (**N_PF**) to be filtered by a function (**F_F**ilter **S**ervice). Sensitive information can be user data or production specifics. Both services are hosted by a common Connector (**C_C**onnector **A**). The filter function removes the sensitive information before exposing it as sanitized data (**D_S**anitized **D**ata). The sanitized data is then forwarded to an analysis function (**F_A**nalysis **S**ervice), hosted at a connector (**C_C**onnector **B**) at the consumer side, in another trust domain.

At the beginning of the analysis we first assign discrete SRL to the starting points. Our scale therefore ranges from level 0 (no relevance) to level 4 (very high relevance) such as in IEC 62443-4-2 [18].

For the confidentiality, we rate the initial data, here from the sensors ($C_{S0-2}$). The confidentiality relevance for the data from $C_{S0}$ is very high ($Conf(D_{S0}) = 4$), from $C_{S1}$ it is moderate ($Conf(D_{S1}) = 2$), and from $C_{S2}$ it is negligible ($Conf(D_{S2}) = 0$).
For the integrity we rate the sinks and the cyber-physical functions. The integrity relevance of the sink **F_A**nalysis **S**ervice ($F_{AS}$) is high ($Int(F_{AS}) = 3$), while the availability relevance is low ($Ava(F_{AS}) = 1$). The integrity relevance of the cyber-physical function **F_P**roduction **S**ervice ($F_{PS}$) is very high ($Int(F_{PS}) = 4$), while the availability relevance is high ($Ava(F_{PS}) = 3$).

A specialty here is the function **F_P**roduction**S**ervice ($F_{PS}$), which adopts the confidentiality rating of the processed data for its integrity ($Int(F_{FS}) = Conf(D_{AG})$. It removes most confidential information from **D_A**ggregated **D**ata ($D_{AG}$), reducing the confidentiality of the then sanitized **D_S**anizited **D**ata ($D_{SD}$) to low ($Conf(D_{SD} = 1)$).

Based on these initial ratings, the SRL are propagated along each particular communication vector (consisting of connections, data, components and functions).

Table 1 demonstrates this propagation process for the example shown in Figure 2.

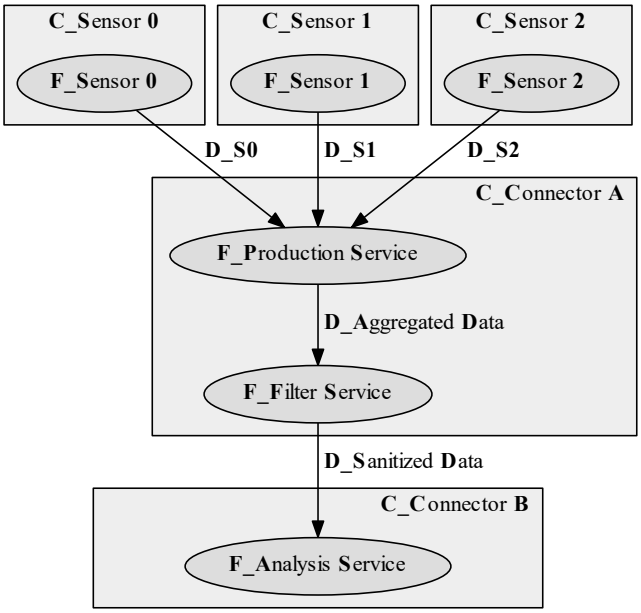


**Figure 2: Use case example.**

| Instance | Confidentiality (Conf) ↓ | Integrity (Int) ↑ | Availability (Ava) ↑ |
|---|---|---|---|
| $D_{S0}$ | **4** | $F_{PS}$ | $F_{PS}$ |
| $F_{S0}$ | $\max(1, D_{S0})$ | $D_{S0}$ | $D_{S0}$ |
| $N_{S0}$ | $D_{S0}$ | $F_{PS}$ | $F_{PS}$ |
| $C_{S0}$ | $F_{S0}$ | $D_{S0}$ | $D_{S0}$ |
| $D_{S1}$ | **2** | $F_{PS}$ | $F_{PS}$ |
| $F_{S1}$ | $\max(1, D_{S1})$ | $D_{S1}$ | $D_{S1}$ |
| $N_{S1}$ | $D_{S1}$ | $F_{PS}$ | $F_{PS}$ |
| $C_{S1}$ | $F_{S1}$ | $D_{S1}$ | $D_{S1}$ |
| $D_{S2}$ | **0** | $F_{PS}$ | $F_{PS}$ |
| $F_{S2}$ | $\max(1, D_{S2})$ | $D_{S2}$ | $D_{S2}$ |
| $N_{S2}$ | $D_{S2}$ | $F_{PS}$ | $F_{PS}$ |
| $C_{S2}$ | $F_{S2}$ | $D_{S2}$ | $D_{S2}$ |
| $F_{PS}$ | $\max(1, D_{S[0-2]})$ | $\max(\mathbf{4}, D_{AG})$ | **3** |
| $D_{AG}$ | $F_{PS}$ | $F_{FS}$ | $F_{FS}$ |
| $N_{PF}$ | $D_{AG}$ | $F_{FS}$ | $F_{FS}$ |
| $F_{FS}$ | $\max(1, D_{AG}, D_{SD})$ | $\max(D_{SD}, CNF(D_{AG}))$ | $D_{SD}$ |
| $D_{SD}$ | **1** (decreased by $F_{FS}$) | $F_{AS}$ | $F_{AS}$ |
| $N_{AB}$ | $D_{SD}$ | $D_{SD}$ | $F_{AS}$ |
| $C_{CA}$ | $\max(F_{PS}, F_{FS})$ | $\max(D_{AG}, F_{PS}, F_{FS})$ | $\max(F_{PS}, F_{FS})$ |
| $F_{AS}$ | $\max(1, D_{SD})$ | **3** | **1** |
| $C_{CB}$ | $F_{AS}$ | $\max(1, F_{AS})$ | $F_{AS}$ |

**Table 1: Security relevance level propagation of the use case example.**

## 4 REQUIREMENT DERIVATION

Based on identified protection needs, requirements are derived. The combination of security goal and class of each instance requires different measures. Based on the respective class and security objective, we provide an enumeration list of suitable security measures in Table 2. Although this enumeration is not exhaustive, it represents a good starting point. Especially for dynamic structures the path from which an SRL was derived should be considered when choosing a protective measure. Advanced measures to monitor security objectives, and the implemented protection measures are subject of Chapter 5.

### 4.1 Security Relevance based Prioritization

To clearly formulate and prioritize security requirements from the determined SRL, we follow the guidelines of RFC 2119 [5] (description of keywords), and RFC 8174 [24] (description of their use in capital letters). The SRL indicates the priority of protection and the need for advanced monitoring measures.

In order to meet the individual security needs of an organization, an individually adapted scale is recommended. Depending on the risk appetite of the respective organization, the decision on the minimal SRL by which a security requirement becomes recommended or mandatory must be considered. This consideration must further

| Objective | Function | Component | Connection | Data |
|---|---|---|---|---|
| Confidentiality | Obfuscation, Runtime protection | Access control, Storage encryption | Channel or message encryption | Encryption |
| Integrity | Software test, Runtime protection | Tamper protection | Firewall | Signature, HSM |
| Availability | Error handling, Redundancy | Timeout, Redundancy | Redundancy, Error handling | Fallback values |

**Table 2: Exemplary protection measures per security objective and class.**

account the individual costs, different effectiveness and side-effects of each protective measure.

For this work, we define that level 0 leads to no individual requirement, level 1 indicates that a protection or monitoring measure SHOULD, while for level 2 some protection MUST be implemented. Level 3 assets MUST have a protection, which SHOULD be monitored, while for level 4, they MUST have a valid protection and an appropriate monitoring measure.

## 4.2 Measures per Security Objective

The type of appropriate security measure depends on the targeted security objective. A brief matching between security objectives and classes is provided in Table 2.

*Confidentiality.* For data, confidentiality can be maintained 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.

*Integrity.* Integrity of data can be ensured by validation [13]. Integrity of connections can be ensured by network monitoring [35]. 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.* 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 measures, while Service Level Agreements (SLA) are administrative ones, especially for functions provided by third-parties.

*Authenticity.* 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 Hardware Security Modules (HSM) to validate their authenticity.

Supplementing these requirements with specific information, rules for network- or host-based monitoring and detection measures can be created and initiated immediately. However, firewall rules and risky intrusion handling measures should only be prepared and proposed to the operator. The information that must be provided by the operator for the relevant entities includes: MAC and IP addresses, used protocols, frequency, data types, and trust boundaries.

## 4.3 Requirements for the Use Case Example

To demonstrate the requirements derivation process, we stick to the example from Section 3.3. Based on the SRL of the instances shown in Table 1, we provide prioritized requirements. Instances without relevance, thus a SRL of 0 have no requirements assigned.

To ensure confidentiality, the following requirements apply: Based on their SRL, $D_{S0}$, $D_{S1}$ and $D_{AG}$ MUST be encrypted, while $D_{SD}$ SHOULD be. Accordingly, channel encryption MUST be implemented by $N_{S0}$ and $N_{S1}$, while it SHOULD be by $N_{AB}$. Furthermore, $N_{PF}$ MUST have its shared memory or messages encrypted. Runtime protection and secure boot MUST be implemented for $F_{S0}$ and $F_{S1}$, while it SHOULD be for $F_{S1}$. An isolation MUST be implemented for $F_{PS}$, $F_{FS}$, while it SHOULD be for $F_{AS}$. Access control MUST be implemented for $C_{S0}$, $C_{S1}$, and $C_{CA}$, while it SHOULD be for $C_{S2}$ and $C_{CB}$.

Furthermore, for integrity, the following requirements apply: All components $C_{S0}$, $C_{S1}$, $C_{S2}$, $C_{CA}$, and $C_{CB}$ MUST be protected against tempering. A network firewall MUST filter the traffic of the connections $N_{S0}$, $N_{S1}$, $N_{S2}$, and $N_{AB}$, while $N_{AB}$ MUST have an application firewall. Signatures MUST be applied and validated for $D_{S0}$, $D_{S1}$, $D_{S2}$, $D_{AG}$, and $D_{SD}$. Software tests MUST be performed for, and runtime protection MUST be applied to $F_{S0}$, $F_{S1}$, $F_{S2}$, $F_{PS}$, $F_{FS}$, and $F_{AS}$.

Finally, the requirements for availability: An error handling MUST be implemented by $F_{S0}$, $F_{S1}$, $F_{S2}$, and $F_{PS}$, while it SHOULD be implemented by $F_{FS}$, and $F_{AS}$. $C_{S0}$, $C_{S1}$, $C_{S2}$, $C_{CA}$, and $C_{CB}$ MUST be implemented redundantly. Fallback values MUST be available for $D_{S0}$, $D_{S1}$, and $D_{S2}$, while they SHOULD be for $D_{AG}$ and $D_{SD}$. A backup connection MUST be available for $N_{S0}$, $N_{S1}$, $N_{S2}$, while it SHOULD be for $N_{AB}$. For the inter-process connection $N_{PF}$, an error handling SHOULD be implemented.

With this structured and conceptualized approach, we derive a possibly large set of measures. Enforcing all these security requirements may be tedious. To ensure their effectiveness, we provide detection methods for them in the following chapter.

# 5 DETECTION AND MONITORING CATALOG

After the security requirements are identified, their continuous fulfillment must be monitored. Additionally, thorough monitoring allows for the identification of new and unknown threats and

attacks. Hence, it is a necessary step for hardening the IT infrastructure for the future use cases of IIoT. A suitable monitoring and detection system includes five major steps:

- data acquisition
- data pre-processing
- data forwarding
- data analysis
- data reporting

Identifying suitable solutions for data acquisition and pre-processing requires an understanding of the different available data types which are relevant in the IIoT scenario.

## 5.1 Data Categorization

In the field of IIoT, there are several data sources which should be monitored. Table 3 lists relevant sources and connects them to suitable monitoring techniques. Data sources can be categorized according to their *location* where the data can be recorded, its *data type*, and the place where its *analysis* should take place. Most data are best recorded locally at each machine participating in the system. Recording locally ensures later analyses are based on the actual data as it arrived or has been measured reducing the risk of intermittent modification. However, as the evaluation of measured data requires significant resources and may depend on other data, remote systems provide interfaces for analysis. While some intrusion detection frameworks already incorporate such distributed setups, a more generic approach for distributed data acquisition in the case of network traffic is given in [16]. In stages prior to the final deployment of the system, output of simulations such as described in [36] give a basis for the implementation and test of suitable analysis mechanisms.

| Data source | Location | Data type | Analysis |
|---|---|---|---|
| executed software | local | measurement | remote |
| application data | local | parse-able format | local |
| log data | local | text format | remote |
| network architecture | global | metadata information | remote |
| network behavior | local | measurement | remote |
| network traffic | local | measurement | remote |
| system behavior | global | measurement | remote |

**Table 3: Data sources relevant in IIoT settings and corresponding monitoring techniques.**

## 5.2 Data Acquisition and Pre-Processing

As pointed out previously, most data available in IIoT scenarios is best measured locally but analyzed at a remote location. Since these scenarios incorporate a lot of different components, an efficient strategy for pre-processing data is required. Considering the different data types available, a set of processing methods is required. While system log files can be transmitted directly, network traffic may be compressed and forwarded using secured connections as described in [16]. Thereby, the legal conformity of data collection and remote storage must be ensured.

Application data which is directly involved in the IIoT process should be subject to an input and output validation [13]. Logs produced by this validation procedure should also be forwarded to the offsite analysis system. The executed software on each system can be measured using cryptographic hashes. Using a hardware trust-anchor and measured boot concepts on the systems, these hashes are protected from manipulation. As the network architecture and system behavior are global properties of the systems, no pre-processing can be done on them but instead the measurement should be considered.

| Data source | Pre-processing |
|---|---|
| executed software | hashes of binaries |
| application data | data validation logs |
| log data | whole log files |
| network architecture | *not applicable* |
| network behavior | listening addresses/ports and associated processes |
| network traffic | compression and forwarding |
| system behavior | *not applicable* |

**Table 4: Local pre-processing of data sources to facilitate and streamline the detection approaches.**

## 5.3 Data Analysis and Reporting

After pre-processing the data, we can transmit it to an offsite analysis system using TLS-secured connections. To ensure a trusted communication, TLS with client and server authentication is required. Depending on the complexity of the network, pre-shared device-specific keys may be used for the authentication. In networks with higher system complexity or dynamics, the use of client and server certificates should be considered.

The analysis system is in charge of checking all defined rules against the data provided by the participating systems. Violated rules lead to an alert which is signaled to the system operator in a Security Information and Event Management system (SIEM). Additionally, the SIEM allows for manual checks of all provided data. Suitable filtering and correlation techniques may be used for advanced manual analyses.

The core of the detection of anomalies and intrusions is, thus, the definition of suitable rules for the data provided.

## 5.4 Deriving Monitoring Rules

Using our lightweight security analysis, we derive a set of requirements as described in Chapter 4. These security requirements can now be translated automatically into rules for an analysis system. Each rule has an alert level which defines the severity of its violation. The impact level of a violation of corresponding protection goals defines this alert level. In the following, for each measure described in Table 2, details and examples are given to automatically derive corresponding monitoring rules.

*Tamper Detection.* The restricted access ensured by corporate production environments yields a physical tamper protection in IIoT scenarios. However, manipulations may still occur even in air-gapped systems. Checking the file integrity on affected components guarantees the detection on any unintended changes. A rule for state-of-the-art host-based intrusion detection systems to check the integrity of a specific file or folder, e.g. a software artifact, or some data asset, is shown below.

```
<syscheck>
```

```
    <directories report_changes="yes"
        realtime="yes" check_all="yes">
        /path/to/folder/or/file
    </directories>
</syscheck>
```

Such a configuration is used locally to derive a list of files and folders to compute hashes on a regular basis or upon changes. As many systems in IIoT scenarios are based on the Linux operating system, a real-time monitoring can be deployed using the `inotify`-subsystem [26]. A whitelisting approach of specific hashes allows for trusted changes in the measured files. Whitelisting hashes of known and trusted files can be done by adding them to a database. With wazuh [40] and OSSEC [1], there are two host-based detection frameworks available with inbuilt whitelist support. While these systems monitor file changes using `md5` and `sha1` hashes, the whitelists only support `md5` hashes which may be considered too weak [39]. Therefore, custom implementations of such logic shall use state-of-the-art cryptographic hash functions.

```
INSERT INTO files (file, md5sum)
VALUES ('/path/to/file',
    '22ee5cebb5bddfad5490633dab7d1afc');
```

By comparing new hashes against those stored in a whitelist database, the offsite analysis system can suppress false positive alerts for intended file changes. A corresponding configuration option for OSSEC/wazuh is shown below.

```
<global>
    <md5db>/path/to/md5.db</md5db>
</global>
```

*Network Monitoring.* The huge amount of network traffic arising in IIoT scenarios may either be analyzed locally or remotely. In the case the device has enough resources, a local processing using state-of-the-art network intrusion detection systems (NIDS), e.g. snort [33] or suricata [38], ensures immediate detection of threats. However, most IIoT scenarios are built upon low-resource devices which profit from a remote analysis. In that case, a local acquisition, lightweight compression, and forwarding as shown in [16] minimizes the impact of the monitoring as much as possible. After transmission, state-of-the-art NIDS perform the analysis on batches of the recorded data.

We define the network behavior for each device as the set of all services and listening ports and addresses of it. Most operating systems provide a suitable mechanism to derive such a list using onboard tools. The output of these tools should be logged locally and reported to the offsite analysis system.

The analysis system must, additionally, monitor the network for its architecture. This includes details about communication partners and paths. As IIoT use cases always incorporate physical sensors and actuators to some degree, a feedback loop between the physical world and its digital counterparts is given. This feedback is at least in parts shared over the network for digital control of the underlying physical processes. Therefore, a suitable system for tracking and analyzing the system behavior is needed. Such systems have been previously described in [7, 14, 15, 30, 32, 35].

*Link Detection.* The availability and configuration of network interfaces can be detected using onboard tools on almost all common operating systems.

*Heartbeat/Timeout.* If a specific service is up and running can be checked locally or from remote systems. Testing the availability of a service is best done by a connection attempt. A failed connection attempt indicates an unresponsive or unreachable service and must be logged.

*Functional Testing.* If a function must be treated as a black box, testing the behavior of the function indicates whether it is working as intended or not. Such a functional testing can be implemented with comparatively low effort using unit tests from software testing. However, specifically tailored tests may produce even more accurate results.

*Encryption of Storage and Data.* While the encryption of transferred data and storage affects two different classes of the previously described model, the measure to protect their confidentiality is the same. As encryption of data and storage essentially provides a binary blob which cannot be checked for its validity, it is hard to monitor if such a requirement is met. However, we can measure the entropy in the binary blob by reading the file or storage medium and calculating the entropy given as

$$E = -\sum_{\alpha \in B} p_\alpha \log_2(p_\alpha) \tag{1}$$

whereas $B$ denotes the set of all possible byte values, i.e. $B = \{0x00, 0x01, \ldots, 0xff\}$ and $p_\alpha$ represents the frequency of occurrence of $\alpha$ in the investigated storage medium. Assuming data is encrypted using state-of-the-art algorithms, the frequency of occurrence from different bytes must be almost identical. If this was not the case, then the encryption algorithm either did not manage to hide the information, which would allow for attacks, or the data was not encrypted at all. Assuming a proper encryption has taken place, for every $\alpha, \beta \in B$ the frequency should be very similar, i.e. $p_\alpha \approx p_\beta$.

Hence, for a theoretical correct encryption Equation 1 simplifies to

$$E^{enc} = \log_2(|B|) \tag{2}$$

whereas in all other cases, the Entropy $E^*$ will be less than $E^{enc}$.

Thus, the measurement of entropy in a file or storage medium yields indicators for how well encryption has been implemented and applied to it.

*Signatures.* While cryptographic signatures are a great tool to provide authenticity for information, they usually require the setup of a public key infrastructure (PKI) in IIoT use cases. If we face a requirement indicating the need of signatures, it is, thus, needed anyway, that all affected components have access to methods to verify the signature. The same method can then be used by the monitoring system. If this method is not available or is not able to verify a signature or there is no signature for the required data or function available, an alert must be logged.

*Trusted Execution Environment.* Specific architectures like Intel's Trusted Execution Technology (TXT) and ARM's TrustZone provide a suitable measure to execute code in a restricted enclave, effectively

sandboxing it from other code on the same processor. Thus, the restriction of shared hardware resources and information transfers prevents many side-channel attacks.

*Hardware Security Modules.* Hardware Security Modules are a possibility to judge a components authenticity as they provide methods to measure the executed code in a physically secured environment. By implementing trusted boot, an HSM can provide hashes of executed software beginning with the boot phase up to software executed in user space. Any modification to the initial setup of a component can, thus, be detected. Apart of this component integrity measurement, HSMs also provide capabilities for storing identification material such as private key material of certificates in a physically secured space.

*Data Validation.* Data transported in IIoT use cases is consumed or produced by some function as described in the model (cf. Chapter 3.1). Thus, data relates to general input and output in software engineering. A possible solution for monitoring the validity of such inputs and outputs is, therefore, applying the same measures. Recent work [13] showed the effectiveness of checking the input validity to functions in industrial use cases. However, these methods must be adapted to each data type and function.

Finally, the described setup results in a loose architecture as shown in Figure 3. There is no need to require a certain network topology as long as each component may reach the offsite analysis system.

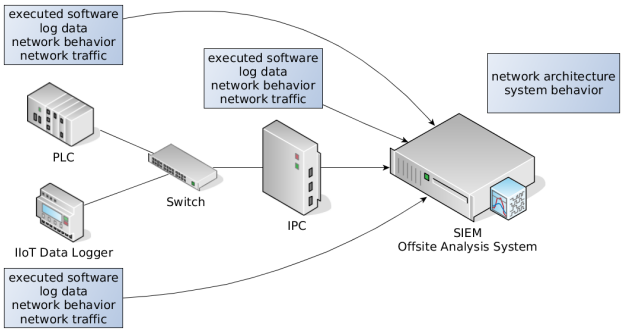


**Figure 3: Architecture for holistic monitoring and intrusion detection.**

## 6 CONCLUSION

In this work, we devised an interactive cybersecurity framework to identify valuable assets and address their protection needs. Our framework supports the operator in deriving security requirements for cyber-physical systems in the Industrial Internet of Things, selecting suitable protective measures, and monitoring their actual effectiveness. To this end, we introduced a lightweight formal modeling approach that allows us to conduct an impact assessment at the design level using a simple set of propagation rules. The result of this step is a list of security relevance levels from which we derive security requirements independent of their implementation. To facilitate the implementation of the generated requirements, we provided recommendations for protection measures and a catalog of advanced monitoring measures tailored to the IIoT domain. This catalog enables a validation of the protection measures and provides information about the actual security state. By linking the catalog with requirements from the analysis, we created a framework that dynamically derives security requirements, technical recommendations for implementation and methods for their continuous validation. We illustrated the use of the framework with a running example that is inspired by real-world applications. Finally, the framework enables the derivation of rules for current solutions for the detection of anomalies. In this way, we support the often-cumbersome definition and prioritization of monitoring rules through an impact-based automated approach.

## ACKNOWLEDGMENTS

Parts of this work were funded by the German Federal Ministry of Education and Research (BMBF) as part of the IUNO Insec project under grant № 16KIS0933K.

## REFERENCES

[1] Ali Anafcheh et al. 2018. Intrusion Detection with OSSEC.
[2] Daniel Angermeier, Alexander Nieding, and Jörn Eichler. 2016. Systematic Identification of Security Goals and Threats in Risk Assessment. *Softwaretechnik-Trends* 36, 3 (2016).
[3] Simon Duque Antón, Daniel Fraunholz, Christoph Lipps, Frederic Pohl, Marc Zimmermann, and Hans D. Schotten. 2017. Two decades of SCADA exploitation: A brief history. In *IEEE Conference on Application, Information and Network Security (AINS '17)*. IEEE, Piscataway, NJ, USA, 98–104. https://doi.org/10.1109/AINS.2017.8270432
[4] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. 2012. *skywiper (aka flame aka flamer): A complex malware for targeted attacks.* Technical Report. Laboratory of Cryptography and System Security (CrySyS) Budapest University of Technology and Economics.
[5] Scott Bradner. 1997. *IETF RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.* Technical Report. Internet Engineering Task Force (IETF).
[6] Gerd S. Brost, Manuel Huber, Michael Weiß, Mykolai Protsenko, Julian Schütte, and Sascha Wessel. 2018. An Ecosystem and IoT Device Architecture for Building Trust in the Industrial Data Space. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security - CPSS '18*, Dieter Gollmann and Jianying Zhou (Eds.). ACM, New York, NY, USA, 39–50. https://doi.org/10.1145/3198458.3198459
[7] Marco Caselli, Emmanuele Zambon, and Frank Kargl. 2015. Sequence-aware Intrusion Detection in Industrial Control Systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS '15)*. ACM, New York, NY, USA, 13–24. https://doi.org/10.1145/2732198.2732200
[8] Thomas M Chen. 2010. Stuxnet, the real start of cyber warfare?[Editor's Note]. *IEEE Network* 24, 6 (2010), 2–3. https://doi.org/10.1109/MNET.2010.5634434
[9] Common Criteria. 2017. *Common Methodology for Information Technology Security Evaluation: Evaluation Methodology* (3.1r5 ed.). Standard. Common Criteria. https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
[10] Deutsches Institut für Normung e. V. 2016. *DIN SPEC 91345, Reference Architecture Model Industrie 4.0 (RAMI4.0).* Standard. DIN, Berlin, Germany.
[11] Benjamin Fabian, Seda Gürses, Maritta Heisel, Thomas Santen, and Holger Schmidt. 2010. A comparison of security requirements engineering methods. *Requirements Engineering* 15, 1 (2010), 7–40. https://doi.org/10.1007/s00766-009-0092-x
[12] William Michael Fitzgerald. 2010. *An Ontology Engineering Approach To Network Access Control Configuration.* Ph.D. Dissertation. University College Cork, Ireland, CORK.
[13] Alexander Giehl and Norbert Wiedermann. 2018. Security Verification of Third Party Design Files in Manufacturing. In *Proceedings of the 2018 10th International Conference on Computer and Automation Engineering (ICCAE 2018)*. ACM, New York, NY, USA, 166–173. https://doi.org/10.1145/3192975.3192984
[14] Dina Hadžiosmanović, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. 2014. Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC '14)*. ACM, New York, NY, USA, 126–135. https://doi.org/10.1145/2664243.2664277
[15] Piroska Haller and Béla Genge. 2017. Using sensitivity analysis and cross-association for the design of intrusion detection systems in industrial cyber-physical systems. *IEEE Access* 5 (2017), 9336–9347. https://doi.org/10.1109/

ACCESS.2017.2703906

[16] Gerhard Hansch, Peter Schneider, and Sven Plaga. 2017. 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)*, Vol. 1. IEEE, Piscataway, NJ, USA, 66–70. https://doi.org/10.1109/IDAACS.2017.8095051

[17] Industrial Internet Consortium. 2017. *The Industrial Internet of Things Volume G1: Reference Architecture* (1.80 ed.). Standard. IIC.

[18] International Electrotechnical Commission and others. 2019. *IEC 62443-4-2:2019, Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components.* Standard. IEC, Geneva, CH. 192 pages.

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

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

[21] Nasser Jazdi. 2014. Cyber physical systems in the context of Industry 4.0. In *IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR '14)*. IEEE, Cluj-Napoca, Romania, 1–4. https://doi.org/10.1109/AQTR.2014.6857843

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

[23] Ralph Langner. 2013. *To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve.* Report. The Langner Group, Hamburg, Germany.

[24] Barry Leiba. 2017. *IETF RFC 8174: Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words.* Technical Report. Internet Engineering Task Force (IETF).

[25] Shi-Wan Lin, Brett Murphy, Erich Clauer, Ulrich Loewen, Ralf Neubert, Gerd Bachmann, Madhusudan Pai, and Martin Hank. 2017. Architecture Alignment and Interoperability - An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper. https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf

[26] Robert Love. 2005. Kernel korner: Intro to inotify. *Linux Journal* 2005, 139 (2005), 8.

[27] Mark Nicolett and Kelly M Kavanagh. 2011. *Magic Quadrant for Security Information and Event Management.* Gartner RAS Core Research Note G00212454. Gartner Research.

[28] Boris Otto, Steffen Lohmann, Sören Auer, Gerd S. Brost, Jan Cirullies, Andreas Eitel, Thilo Ernst, Christian Haas, Manuel Huber, Christian Jung, et al. 2017. *Reference architecture model for the Industrial Data Space.* Technical Report. Fraunhofer-Gesellschaft, Munich.

[29] Xinming Ou. 2005. *A logic-programming approach to network security analysis.* Ph.D. Dissertation. Princeton University Princeton. http://worldcatlibraries.org/wcpa/oclc/62317755

[30] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. 2013. Attack detection and identification in cyber-physical systems. *IEEE Trans. Automat. Control* 58, 11 (2013), 2715–2729. https://doi.org/10.1109/TAC.2013.2266831

[31] Florian Patzer, Ankush Meshram, Pascal Birnstill, Christian Haas, and Jürgen Beyerer. 2018. Towards Computer-aided Security Life Cycle Management for Critical Systems. In *13th International Conference on Critical Information Infrastructures Security (CRITIS '18)*. Springer, Cham, Germany, 45–56. https://doi.org/10.1007/978-3-030-05849-4_4

[32] Sasanka Potluri, Christian Diedrich, and Girish Kumar Reddy Sangala. 2017. 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)*. IEEE, IEEE, Piscataway, NJ, USA, 1–8. https://doi.org/10.1109/ETFA.2017.8247663

[33] Martin Roesch. 1999. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration (LISA '99)*. USENIX Association, Berkeley, CA, USA, 229–238. http://dl.acm.org/citation.cfm?id=1039834.1039864

[34] Karen Scarfone. 2018. *Draft NISTIR 8228: Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks.* Standard. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8228-draft

[35] Peter Schneider and Konstantin Böttinger. 2018. 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)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3264888.3264890

[36] Peter Schneider and Alexander Giehl. 2018. Realistic Data Generation for Anomaly Detection in Industrial Settings using Simulations. In *Computer Security*. Springer, Cham, Germany, 119–134. https://doi.org/10.1007/978-3-030-12786-2_8

[37] Julian Schuette and Gerd S. Brost. 2018. LUCON: Data Flow Control for Message-Based IoT Systems. In *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE '18)*. IEEE, Piscataway, NJ, USA, 289–299. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00052

[38] Suricata. 2018. The Next Generation Intrusion Detection System. https://suricata-ids.org/

[39] Xiaoyun Wang and Hongbo Yu. 2005. How to break MD5 and other hash functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '05)*. Springer, Berlin, Heidelberg, Germany, 19–35. https://doi.org/10.1007/11426639_2

[40] Wazuh Inc. 2015. wazuh. https://wazuh.com/

[41] Jan Wolf, Felix Wieczorek, Frank Schiller, Gerhard Hansch, Norbert Wiedermann, and Martin Hutle. 2016. Adaptive Modelling for Security Analysis of Networked Control Systems. In *4th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR '16)*. BCS Learning & Development, Belfast, UK, 64–73. https://doi.org/10.14236/ewic/ICS2016.8

[42] Marko Wolf and Michael Scheibel. 2012. A systematic approach to a qualified security risk analysis for vehicular IT systems. In *Automotive - Safety & Security 2012*, Erhard Plödereder, Peter Dencker, Herbert Klenk, Hubert B. Keller, and Silke Spitzer (Eds.). Gesellschaft für Informatik e.V., Bonn, Germany, 195–210.