

# Risk-driven Model-based Architecture Design for Secure Information Flows in Manufacturing Infrastructures

Loris Dal Lago<sup>a</sup>, Fabio Federici<sup>b</sup>, Davide Martintoni<sup>c</sup> and Valerio Senni<sup>d</sup>  
*Applied Research and Technology, Collins Aerospace, Piazza dell'Indipendenza 23, 00185 Rome, Italy*

**Keywords:** Risk Assessment, Information-flow Security, Model-based Design, Industrial Manufacturing Security.

**Abstract:** Modern manufacturing infrastructures leverage internet and intranet connectivity to guarantee the remote execution of services at the shopfloor level, continued operations and remote reconfigurability. Nonetheless, equipment used in industrial plants is not always prepared to withstand the security challenges introduced by increased connectivity demands, thus exposing the overall system to security threats. We propose a model-based approach to combine secure design of digital infrastructures for manufacturing with a rigorous security risk assessment, enabling trusted connectivity for equipment, with a robust analysis method for the evaluation of their security properties. To that aim, information flow paths are captured between functions and equipment, assets and threats are identified, mitigations and new security requirements are defined. Mitigations are then propagated to the level of implementation, where we rely on hardware-enforced isolation to provide trusted computation and data protection. In this paper we demonstrate our methodological approach using an extension of the SysML language for threat modelling and by relying on ARM TrustZone for hardware isolation. Our approach is sufficiently general to be reused for other domains and alternative technologies.


## 1 INTRODUCTION


In the age of IoT and connected infrastructures, the security of systems is of paramount importance for the availability of services and the reputation of companies providing those services. The domain of digital manufacturing has started to focus, more and more, on remote collaboration solutions, rising the challenge of new connectivity demands and entailing strong obligations to secure data produced and stored by manufacturing equipment. At the level of shop floors, information is essential to optimize operations, guarantee products quality, support innovation and reduce costs related to certification (Int. Data Spaces Assn., 2021). Moreover, manufacturing solutions for safety-critical products (e.g., for automotive or aerospace application domains) require a high level of security assurance, to establish trust in the supply chain and ensure the safety of the end systems. In this domain, it is common to use third-party's equipment, whose detailed functions and security configurations


may be only partially available. It is well known, on the other hand, that the endpoints of a network infrastructure are frequently the weakest nodes, due to their limited computational capabilities and inevitable presence of legacy, thus becoming easy vectors for broader attacks (Dragos, 2021).


Motivated by these considerations, we propose a systematic approach that (1) combines the design of digital manufacturing infrastructures with their security risk assessment, by identifying current threats and evaluating the level of risk associated with them, (2) supports the identification of mitigations to the found security threats, (3) helps tracing the security elements to the architectural components they relate to and (4) guides the implementation of security measures, with focus on the protection of remotely accessible endpoints.

As a first step, our approach leverages model-based representations of architectures to analyze information flows and security policies, support the identification of weaknesses, quantify their induced

<sup>a</sup>  <https://orcid.org/0000-0001-6098-1793>

<sup>b</sup>  <https://orcid.org/0000-0003-4179-3986>

<sup>c</sup>  <https://orcid.org/0000-0002-1648-0104>

<sup>d</sup>  <https://orcid.org/0000-0002-1131-0384>

risks, and propose mitigations for the unacceptable ones. Mitigations induce new security requirements and their allocation to elements of the architecture.

The fulfillment of the new security requirements is then supported by providing guidance and best practices for the addition of *Architectural Patterns* (i.e. architectural solutions for the enforcement of security properties) and the use of *Security Building Blocks* (i.e. SW/HW implementation solutions that can be used to realize specific security functions). In the case of reference we will introduce a gateway as an Architectural Pattern to manage the separation of information flows of legacy endpoint equipments. We will further use SW routines on top of hardware-enforced isolation primitives as the Security Building Blocks, allowing to protect edge components and their data during the execution of sensitive processes.

To facilitate adoption and minimize disruption of existing processes, our approach is developed on top of well-recognized modeling tools/languages and established risk analysis practices. We present the evaluation of the approach on a remote maintenance case study, where components of different parties and with different levels of trust access share functions within the boundaries of the same architecture.

The paper is structured as follows. Section 2 describes our security risk assessment methodology. Section 3 provides details on the use of SysML models for risk assessments. Section 4 showcases the application of our method to a case study inspired by the manufacturing domain, with details on how the required mitigations are implemented. Finally, we discuss related work and future directions.

## 2 SECURITY RISK ASSESSMENT

The literature offers several approaches to cybersecurity risk assessment (Rocchetto, Ferrari, & Senni, 2019) and their terminology vary from one approach to the other, based on the domain and community. Our terminology and workflow are inspired by the cybersecurity risk assessment guidelines of DO-326A/DO-356A aerospace standards, whose guiding principles are sufficiently general to be applied to domains other than aerospace. With these premises, we start our discussion by defining the key terms used in the rest of the paper:

**Asset:** a valuable logical or physical resource that requires protection from a security perspective.

**Security Policy:** the rules to determine the allowed security interactions for a system, including the identification of trusted flows of information from assets to internal and external entities.

**Attack Vector:** an exploitable entry point or interface on the system architecture for external malicious actors to perform an attack.

**Threat Condition:** an unwanted condition (or state) of an *Asset*, caused by the activity of external malicious actors, with consequences to the security of the overall system.

**Threat Model:** a characterization of a set of *Threat Conditions* in terms of target *Assets*, evaluated against envisioned activities of external malicious actors and their accessibility to *Attack Vectors*.

**Threat Scenario:** a description of a potential unauthorized interaction by an external malicious actor, identifying the *Attack Vector* and the path in the architecture leading to a *Threat Condition*.

**Risk:** a quantitative account of the *Threat Conditions* associated to a specific *Asset*, in terms of severity and exposure to the identified *Threat Scenarios*.

**Risk Evaluation:** the analysis of all *Threat Conditions* affecting an *Asset* to quantify the associated *Risk*.

**Risk Mitigation:** the quantitative reduction of a *Risk* by modification of an architecture to reduce the impact, severity or occurrence of *Threat Scenarios*.

**Security Measure:** The response to a *Risk Evaluation*, identified by activities of *Risk Mitigation*.

The methodology and high-level workflow to the security risk assessment of digital manufacturing architectures is provided in Figure 1.

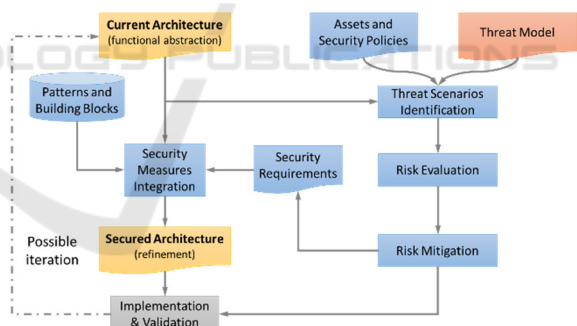


Figure 1: Methodology for risk assessment.

On the left of the picture, we represent the desired improvement of the *Current* (potentially insecure) *Architecture* to a final *Secured Architecture*, achieved through the integration of appropriate *Security Measures*. The initial version of the *Current Architecture* is a preliminary description of the system, representing either a legacy infrastructure, a new design, or a mix between the two. Improvement of this architecture may, in some cases, require a few iterations, until all appropriate mitigations are in place. In this paper the *Current Architecture* is

assumed to be available, and we defer to future work the challenges associated to creating one.

On the right of the picture, we represent the process through which the *Security Measures* are derived, based on the *Security Requirements* yielded by the workflow. The key inputs to this process are *Assets*, *Security Policies*, and the *Threat Model*. Those inputs, in conjunction with the *Current Architecture*, are used to identify the *Threat Scenarios*, which are described in terms of elements of the *Current Architecture*. Then, *Threat Scenarios* are analysed to provide a *Risk Evaluation*. This step is performed by aggregating *Threat Scenarios* by their cumulative impact on *Assets* and considering their realization complexity. This leads to the systematic identification of potential weaknesses in the *Current Architecture* and a characterization of information flows, which may expose the overall system to the occurrence of *Threat Conditions*. Thus, the final step consists in the selection of the above-threshold (unacceptable) risks and identification of appropriate *Risk Mitigations* to address those risks.

The introduction of Risk Mitigations facilitates the generation of new *Security Requirements*, yielding the refinement of the *Current Architecture* to the *Secured Architecture*. Driven by the experience of the architect, security requirements may lead to the reorganization of the information flows, the introduction of additional (security) functions and the encapsulation of existing ones in protected context. *Architectural Patterns* and *Security Building Blocks* are considered at this stage, as defined in the introduction, to support the refinement process.

At this point, the *Secured Architecture* will drive the implementation. Being the implementation more concrete, it may hide additional weaknesses and influence security at the level of multiple *Risks*. Therefore, to prove the efficacy of mitigations after the implementation, the identified *Security Measures* and their integration are validated. In case this validation phase finds weaknesses at this level, further iterations and analyses are induced with feedback to the *Current Architecture*, and additional security improvements are provided by the workflow.

### 3 MODEL-BASED SECRA

To realize the Security Risk Assessment (SecRA) workflow presented in Section 2, we present a model-based approach. We base this work on the SysML modelling language, extended with a specific profile for supporting the workflow and analysis. This profile supports the organization of security concepts on top

of the system architecture, as well as their traceability to system artifacts and associated requirements.

To that aim, components are represented by SysML *Blocks*, instantiated as so-called *Parts* in *Internal Block Diagrams*, and connected by their *Ports* via *Connectors*. SysML also provides the native notion of *Requirements*, which we split between *Customer Requirements* and *System Requirements*. Both types of requirements can be linked to architectural components via *allocation* relationships. This is a very typical and general infrastructure in SysML modelling, which we employ in our remote maintenance case study to support the security view.

In the following section, we shall further use the concept of SysML *stereotypes*, which are language primitives to define custom language extensions, by allowing the definition of a meta-model on top of the existing SysML constructs. The elements of the new meta-model are endowed with specific meaning and the so-called *tags* to define their properties. More information about SysML and its constructs can be found in the related standard (OMG, 2019).

#### 3.1 Language Extension for Security

A selected portion of our meta-model, showing the most relevant tags and the relationships between elements, is represented in Figure 2. The lower part of the figure (green) represents the model elements of the *Current* and *Secure Architectures*, as defined in Section 2. They include *Customer* and *System* (Security) *Requirements*, allocated to architectural components, implementing traceability relationships to provide assurance evidence in case of certification or regulatory compliance. As represented on the rightmost part, the requirements are fed by the *Risk Mitigations* identified as an outcome of the Security Risk Assessment workflow defined in Section 2.

The upper part of Figure 2 (yellow) represents the security aspects of our methodology. In particular:

*Assets* – having a tag named “Intangible Asset Protection” (IAP), which specifies their impact to immaterial KPIs such as *Company Reputation*, *Product Quality*, *Intellectual Property*, and others. Assets also have “Kinds”, indicating whether they represent *Logical Data*, *Physical Interfaces*, *Software*, *Storage* or *Logical State*.

*Threat Conditions* – with a tag named “Impact On”, used to indicate whether the threat condition impacts *Confidentiality*, *Integrity* or *Availability* of the related asset. The “Operational Phase” relates the threat condition to system life-cycle or behavioural phases, such as *Installation*, *Maintenance*, *Dismissal*, *Nominal*, *Transient*, *Faulty*. Finally, the “Attack

Severity” tag indicates the effect of the attack, when successful, with values such as *Low*, *Medium*, *High*, and *Critical*. The “Involved Asset” relation indicates the asset involved in the threat condition.

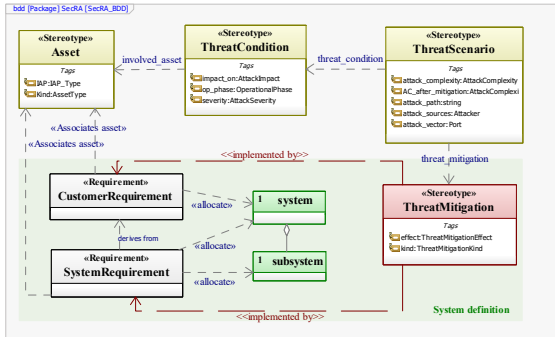


Figure 2: SysML meta-model of the security extension.

**Threat Scenarios** – with two tags associated to “Attack Complexity”, populated before and after the identification of risk mitigations, respectively. In the initial threat scenario, a mitigation might not be available and the only attack complexity under analysis is the default one. When a mitigation is identified, the attack complexity is updated accordingly with a new, higher value. Values available for this tag are *Low*, *Medium*, *High* and *VeryHigh*. The tag called “Attack path” is (currently) a textual description of the scenario. The tag “Attack sources” is populated from a user-defined and system-specific set of SysML actors, whereas the “Attack Vectors” are identified as the ports on the system model. Finally, the “Threat Condition” and the “Risk mitigation” relations indicate, respectively, the threat conditions involved in a specific threat scenario and the and the mitigations in place to reduce its impact or complexity. For any new mitigation we expect an update of the “Attack Complexity” tags.

**Risk Mitigations** - are provided with a tag called “Effect”, specifying whether the mitigation is supposed to have *Preventive*, *Detective*, *Corrective* or *Deterrent* role on the security hardening of the design. The “Kind” indicates whether a mitigation is to be implemented by *Technical* means, *Process* means or by means of a *Guideline*. Finally, the “Implemented By” relation indicates the requirements involved in the fulfilment of a mitigation. This relationship is populated with links to security requirements, which stem out of mitigations identified by our workflow.

We note that in our security language extension we prefer to use abstract values (such as *Low*, *High*, ...) over definite numbers or number ranges representing specific quantities. On the one hand, this allows to

deal with simpler concepts, not requiring deep levels of detail to be available during design. On the other hand, this also keeps the space of possibilities bounded to a limited size and allows to perform the security assessment at a higher level of granularity.

The linkage between top and bottom areas of Figure 2 is captured by the relationships of the system and customer requirements with the security assets (on the input side) and with the risk mitigations (on the output side). In practice, this allows the realization of the workflow of Section 2 and the execution of the security risk assessment activities in parallel to the system definition. Thus, our approach enables (i) traceability between components, (ii) easy navigation from one artifact to another in the model and (iii) the representation of the impact of one artifact to the rest of the system and its components.

### 3.2 From Modelling to Risk Evaluation, Mitigation, and Validation

With all relevant security aspects of the system captured as part of the model, the *Risk Evaluation* task can be executed. Automated tools extensions are used to navigate the model, extract information, and quantify *Risks* by a user-defined multi-objective *Risk Evaluation Function* (not shown in our meta-model). For each of the *Threat Scenarios*, the *Risk* is computed by considering attack severity, attack impact and attack complexity scores (Rocchetto, Ferrari, & Senni, 2019). Thus, *Threat Scenarios* can be ranked by their *Risk*, and those whose value is above a user-defined acceptable thresholds are identified. Risks can further be combined per-*Asset* or per-*Threat-Condition*, to study their cumulative impact. By analysing commonalities between *Threat Scenarios* associated to the same *Risk*, architectural weaknesses causing multiple *Threat Scenarios* are also identified. In a longer-term vision, this will enable trade-off analyses and study alternative mitigation strategies, where cost, risk-reduction, architecture impact, and new architecture design needs are all considered.

It is worth noting that, since a full-blown behavioural system model is generally unattainable due to the existence of legacy and third-party components in the manufacturing environment, we perform the analysis and derive appropriate mitigations by only using information available to engineers at a sufficiently high level, without creating separate behavioural representations of the security policies or the threat model. Although this step may become useful once the architecture is consolidated, we do not see, for now, the need for that enrichment.



As a final activity, the methodology presented in Section 2 envisions a validation phase, with the intent of confirming effectiveness of mitigations after implementation. After this phase, a reassessment of the security risks may be required, and additional mitigations may be introduced. The model and the traceability relationships support consistency, enable the insertion of new mitigations and a quick efficient re-assessment of the architecture security. So far, we have only performed limited sets of validations on our implemented reference example. We have not yet exercised the full approach at-scale.

## 4 REMOTE MAINTENANCE

The vision of Industry 4.0 revolves around connectivity, edge intelligence, and data integration. Two of the major obstacles to the realization of this vision are (1) the cybersecurity risks inevitably introduced by increased connectivity, and (2) the need of retrofitting legacy equipment, unable to offer the requested capabilities.

Frequently, Industry 4.0 infrastructures are structured into layers, where Operation Technologies (OT) and Information Technologies (IT) reside in separate networks. The OT segment typically hosts all the manufacturing equipment, while the IT segment contains all the enterprise services (engineering, quality, business, process monitoring and optimization). This infrastructure, inspired by the Purdue architectural model (Williams, 1994), divides the IT layer into three sublayers: one dedicated to the manufacturing plant operations, one dedicated to the global and cross-plant operations and another, external, where stakeholders of the overall manufacturing ecosystem can collaborate. Partitioning of each layer into segments is typical, to mitigate the impact of potential attacks (through lateral movement). Also, communication across segments and with the external world is regulated through firewalls, with stringent authorization rules. The IEC organization defined the 62443 family of standards (Schneider Electric, 2018) to provide guidance in applying cybersecurity best-practices to manufacturing infrastructures. The core notions of security promoted by the 62443 are those of *Zones* and *Conduits*. The first notion consists in grouping assets with similar security requirements and enforcing boundaries of protection with the external world. The second notion consists in identifying communication paths between zones with different security requirements and enforcing controls on access and exchanged information.

In this section we consider a practical example of remote maintenance and discuss some of the challenges to provide access to the equipment of the plant while enforcing suitable security policies (in accordance with the notions of zones and conduits). The security requirements and mitigations identified during the security risk assessment determine how the information flows at different levels shall be managed securely. The use of an *Industrial Gateway* as an architectural pattern is a common practice in manufacturing - e.g., to guarantee interoperability among different standards or push computations closer to the edge. Here we are extending its application to the enforcement of conduit security requirements, to provide confidentiality and integrity of manufacturing data and computations.

As a reference throughout the section, Figure 3 represents the remote maintenance case study. The orange box represents the boundary of the risk analysis. It includes endpoint industrial controllers (subject to maintenance) and an intermediary gateway. In the Secured Architecture the gateway hosts the security measures required to regulate the flow of information to the controllers. Information flows may be categorized into “low-security”, such as periodic maintenance, or “high-security” such as industrial controllers firmware/configuration updates.

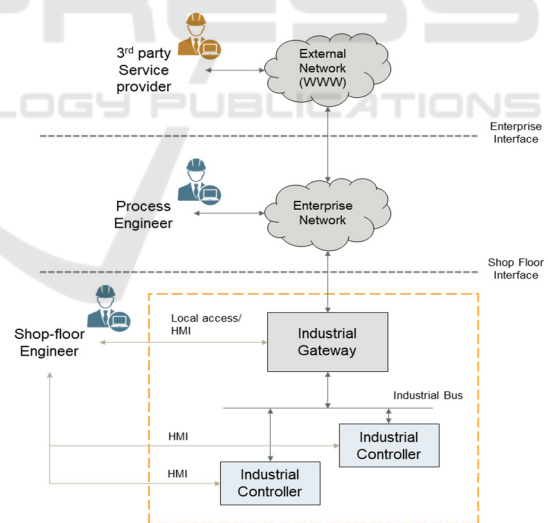


Figure 3: Remote maintenance case study.

The following actors will need interact with the shop-floor devices: (i) the *shop floor engineer*, in charge of routine operation of industrial controllers, (ii) the *process engineer*, in charge of process efficiency and manufacturing quality control by remote interaction with the shop floor devices, and (iii) the *third-party service provider*, in charge of executing (through remote access) maintenance

services, such as execution of diagnostics or firmware updates of shop-floor devices.

The main security requirement of the case study is the runtime protection of sensitive data and computation assets, such as communication interfaces connecting the gateway and the controllers and confidential data stored on the gateway, avoiding unintended interference of third-party activities.

#### 4.1 Method Application

We applied the methodology and modelling approach described in Section 3 to the remote maintenance case study. As a first step, we developed a SysML model (Current Architecture) including actors (shopfloor and process engineer, third-party service provider) and main architectural components (industrial gateway and industrial controllers). The information flows traversing the industrial gateway are categorized as: (1) Controllers Diagnostics, (2) Engineering and Production Data, (3) Controllers Configuration. We also identified Assets, including *Diagnostic Data* (whose integrity is needed for plant operations), *Engineering or Production Data* (subject to confidentiality for IP protection), *Configuration of SW* (subject to integrity to guarantee product quality), and *Logs* (essential for auditing and security forensics). In addition, we captured the *Security Policies*, specifying what is allowed/forbidden (e.g., the flow of Production Data to a third-party service provider not being allowed). Violation of policies may cause a damage to one or more assets, realizing a Threat Condition. Those are captured referring to Assets and potential events, including (e.g.) “Diagnostic Data Integrity Compromise”, “Engineering or Production Data Leaked outside the company boundaries”, “Configuration of SW Integrity compromised during Maintenance”. As prescribed by our meta-model, a Threat Condition has an associated security impact (Confidentiality, Integrity, or Availability) and severity (Low to Critical). The Threat Model is also relevant, as it describes the entry points used by malicious external actors, as well as their target Assets.

All information available in the model facilitates the comprehensive identification of Threat Scenarios, which describe attack paths (e.g., “An internal user logs into the gateway as a normal user, bypasses the access permission mechanism by exploiting unforeseen information-flow paths”). As prescribed in our meta-model, Threat Scenarios have an associated Attack Complexity, ranging from VeryHigh to Low. Threat Scenarios were documented in a table within the SysML model and

automatically ranked, using a simple Risk Evaluation Function defined as a sum of values associated with the severity, impact, and complexity scores. All Threat Scenarios with a Risk Evaluation higher than a critical threshold were selected for mitigation.

The identification of appropriate mitigations was driven by the analysis of all the critical Threat Scenarios. Traceability to model elements supported the identification of some critical sub-functions that are shared across information flows, subject to different Security Policies. As an example, different information flows share the common need of storing data or SW and retrieving it for later use. Similarly, both Engineering/Production and Maintenance flows require access to critical configuration APIs of the controllers, but with different authorizations. Sharing such functions/interfaces exposes the system to the risk of violation of security policies. For this reason, the Risk Mitigation activity led to the identification of highly trusted core functions, such as isolation of process execution and diagnostic data protection.

Security requirements are defined and linked to those functions: e.g., (r1) Diagnostic and Engineering Data shall be stored in isolated environments with non-by-passable access control, (r2) Configuration APIs access shall be isolated from the main (untrusted) execution environment. At the software architecture level (shown in Figure 4), sensitive data and functions were marked as *trusted* and placed in an *isolated execution environment* endowed with implementation primitives that we could selectively use per our methodology as *Security Building Blocks*.

#### 4.2 Platform Selection

For the enforcement of isolation between the trusted components of our reference example, we selected a Trusted Execution Environment (TEE) technology. TEEs are secure domains of computing platforms that achieve high-assurance isolation by exploiting hardware-enforced separation and access control. In this manner, sensitive data and functions can be stored, processed, and protected to different levels of security. The use of TEEs guarantees (1) Isolated execution, (2) Secure storage, (3) Local and remote Attestation, (4) Secure provisioning and (5) Trusted path (Vasudevan & et. al, 2012). TEEs offer protection against software attacks generated in Rich Execution Environments (REEs), that are full-fledged environments with less reliable security protection.

To prototype our implementation, we targeted an Arm hardware architecture, including Arm TrustZone as a TEE. Since Arm v6 architecture (Pinto & Santos, 2019), the TrustZone technology introduced two

isolated protection domains (worlds) that the Arm processors operate in, called *secure world* and *normal world*. We used this technology to run tasks at different levels of trust (and support the isolation of information flows), as well as to protect data.

At firmware level, we considered the OP-TEE framework as a mature supporting solution. OP-TEE is an open-source framework implementing a TEE on top of Arm TrustZone for the Arm Cortex-A family. OP-TEE provides an open-source implementation of the GlobalPlatform API, currently one of the most mature industrial TEE specifications.

The architectural model of our case study has been used to derive (manually, for the moment) the code structure on top of the OP-TEE runtime. Trusted components were associated to the TEE primitive of Trusted Applications, as shown in the SW architecture in Figure 4. Related assets were mapped to data items associated to TEE secure storage. Non-trusted components (e.g., clients for external interactions) were associated to the primitives of the REE (Linux), to ensure they did not interfere with the primitives of the TEE. Finally, ports and connectors in our architecture were associated to inter-process communication primitives and those between trusted and non-trusted components associated to OP-TEE service calls to uphold the separation between them.

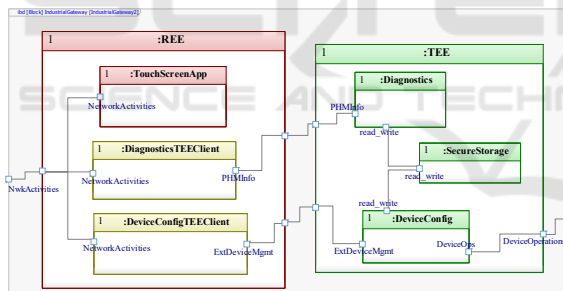


Figure 4: Industrial Gateway software architecture.

## 5 RELATED WORK

In recent years, the concept of secure remote maintenance has been explored in various works. (Schwarz & et. al, 2020) propose a hardware-based solution to isolate sensitive segments of the network from the broad internet, exploiting hardware separation between the external maintainer and the maintenance target. (Kern & Anderl, 2019) exploits a software defined network (SDN) design, combined with an attribute-based policy framework, to implement secure network management during remote maintenance sessions. (Kasinathan & et al.,

2021) apply a workflow-based approach to model and enforce access-control on a remote maintenance scenario. From a security standpoint, those solutions focus mostly on network and access control. Our approach, instead, aims at providing information flow security enforcement by design, at the level of the system security architecture, exploiting industrial gateways at the edge and trusted implementation building blocks, thus being complementary to those.

The literature also offers several graphical and model-based approaches to support security assessments (Jürjens, 2002), (Roudier, 2015), (Gluch, 2019). Although similar in spirit, our approach extends them by proposing an over-encompassing workflow that aims at deriving implementations that can be readily assessed realizing system-level security policies. The CORAS methodology (Lond & et al., 2011) was the first to link architectural elements with cybersecurity risk assessment. In the current work, we enriched a formal architecture modelling language with security concepts, while CORAS proposed the use of independent modelling artefacts. In previous work (Rocchetto, Ferrari, & Senni, 2019) we investigated techniques for fully automated threat scenarios generation, although relying on significant modelling effort and missing the implementation of security mitigations. (Enoch & et al., 2021) summarizes approaches for security attacks modelling, pointing out the need for hierarchical and compositional approaches, which is one of the motivations that led us to use SysML. In this work we adopt a recognized modelling notation, enabling future automations, and we also support the design and implementation of security mitigations by using a specific security technology (TEEs) and pre-defined security building blocks, reducing end-user effort.

A limited number of works relate information flow security to the design and verification of systems leveraging Trusted Execution Environments. (Gollamudi & Chong, 2016) analysed the use of enclaves to enforce information flow security policies against attackers able to inject arbitrary code outside enclave boundaries. (Gollamudi A. S., 2019) also analysed the problem of implementing distributed application using TEEs. Our approach, instead, supports design trade-offs evaluation as well as end-to-end security design and validation, in the context of a model-driven methodology.

## 6 CONCLUSIONS

In this paper we described a model-based methodology to systematically analyse and improve

the security of manufacturing architectures. The proposed methodology extends typical architecture development processes, enriching the system-level view with a security viewpoint. The methodology guides the analysis of risks and their sources, supporting structured mitigation, using trusted architectural patterns and security building blocks. The approach is effective if executed at the right level of abstraction, supporting frequent re-assessments. We demonstrated the approach on a remote maintenance case study, showing how the identified mitigations led to a sufficiently detailed design, apt to derive a possible implementation. TEE primitives were used as building blocks to realize isolation and protection of sensitive data and functions.

One shortcoming of this approach is manifested by the need of an initial architecture model, which may be unavailable or costly to build for an existing manufacturing infrastructure. We plan to explore the use of automation to extract (parts of) the functional architecture from network logs, including components, assets, and information flows.

Another limitation of our approach is the large space of potential threat conditions and scenarios, due to the number of attributes and values in our meta-model. Even in this case, we believe that the use of automation can help in generating and selectively addressing threat conditions and scenarios.

Finally, although we consider access to the manufacturing network by third parties as a central case study of our application, our method relies on the assumption that the manufacturing architecture itself is designed and managed by a single organization. An interesting extension to our methodology is to use the architectural model to foster security coordination across different organizations. This scenario would require a reformulation of the method, considering collaboration tools and separation of tasks and ownerships, which is out of the scope of this work.

## ACKNOWLEDGEMENTS

This research is funded by the European Union's Horizon 2020 Research and Innovation program COLLABS under grant No. 871518.

## REFERENCES

- Dragos. (2021). *State of Industrial Cybersecurity*. Retrieved from <https://hub.dragos.com/hubfs/Reports/2021-Ponemon-Institute-State-of-Industrial-Cybersecurity-Report.pdf>
- Enoch, S., & et al. (2021). Model-based Cybersecurity Analysis: Past Work and Future Directions. *Annual Reliability and Maintainability Symposium (RAMS)*. IEEE.
- Gluch, D. (2019). AADL Security Annex [Draft]. Carnegie-Mellon U, US.
- Gollamudi, A., & Chong, S. (2016). Automatic enforcement of expressive security policies using enclaves. *ACM SIGPLAN Int Conf on Object-Oriented Progr, Systems, Languages, and Appl (OOPSLA)*. ACM.
- Gollamudi, A. S. (2019). Information flow control for distributed trusted execution environments. *IEEE 32nd Comp Security Foundations Symp*. IEEE.
- Int. Data Spaces Assn. (2021). *Data Sovereignty – Critical Success Factors for the Manufacturing Industry*. Retrieved from [https://internationaldataspaces.org/wp-content/uploads/dlm\\_uploads/IDSA-Position-Paper-Data-Sovereignty%E2%80%93Critical-Success-Factor-for-the-Manufacturing-Industry.pdf](https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Position-Paper-Data-Sovereignty%E2%80%93Critical-Success-Factor-for-the-Manufacturing-Industry.pdf)
- Jürjens, J. (2002). UMLsec: Extending UML for secure systems development. *Int Conf on the Unified Modeling Language*. Springer.
- Kasinathan, P., & et al. (2021). *Secure Remote Maintenance via Workflow-Driven Security Framework*. IEEE Blockchain.
- Kern, A., & Anderl, R. (2019). *Securing Industrial Remote Maintenance Sessions using Software-Defined Networking*. 6th Int Conf on SW Defined Systems (SDS).
- Lond, M., & et al. (2011). *Model-based risk analysis – the CORAS approach*. Springer.
- OMG. (2019). *OMG System Modeling Language v.1.6*. <https://www.omg.org/spec/SysML/>.
- Pinto, S., & Santos, N. (2019). Demystifying arm trustzone: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 1-36.
- Rocchetto, M., Ferrari, A., & Senni, V. (2019). Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems. *Resilience of Cyber-Physical Systems* (pp. 25-47). Springer.
- Roudier, Y. a. (2015). SysML-Sec: A model driven approach for designing safe and secure systems. *3rd Int Conf on Model-Driven Eng and SW Devel (MODELSWARD)*. IEEE.
- Schneider Electric. (2018). *Cybersecurity Assessment – The Most Critical Step to Secure an Industrial Control System*. Retrieved from <https://www.se.com/ww/en/download/document/998-20298472/>
- Schwarz, K., & et al. (2020). *Conception of a Secure Remote Maintenance Procedure Using Dedicated Hardware*. Society for Imaging Science and Technology.
- Vasudevan, A., & et al. (2012). Trustworthy execution on mobile devices: What security properties can my mobile platform give me? *Int Conf on Trust and Trustworthy Computing*. (pp. 159-178).
- Williams, T. J. (1994). The Purdue enterprise reference architecture. *Computers in industry*, Vol 24 (2). p.141-158.