IoT4CPS – Trustworthy IoT for CPS

FFG - ICT of the Future
Project No. 863129

Deliverable D3.2
Guidelines, processes and recommendations for the design of dependable IoT Systems

The IoT4CPS Consortium:
AIT – Austrian Institute of Technology GmbH
AVL – AVL List GmbH
DUK – Donau-Universität Krems
IFAT – Infineon Technologies Austria AG
JKU – JK Universität Linz / Institute for Pervasive Computing
JR – Joanneum Research Forschungsgesellschaft mbH
NOKIA – Nokia Solutions and Networks Österreich GmbH
NXP – NXP Semiconductors Austria GmbH
SBA – SBA Research GmbH
SRFG – Salzburg Research Forschungsgesellschaft
SCCH – Software Competence Center Hagenberg GmbH
SAGÖ – Siemens AG Österreich
TTTech – Auto AG
TTTech – TTTech Computertechnik AG
IAIK – TU Graz / Institute for Applied Information Processing and Communications
ITI – TU Graz / Institute for Technical Informatics
TUW – TU Wien / Institute of Computer Engineering
XNET – X-Net Services GmbH

For more information on this document or the IoT4CPS project, please contact:
Mario Drobics, AIT Austrian Institute of Technology, mario.drobics@ait.ac.at

Document Control

Title: Design & Methods Concept
Type: Public
Editor(s): Stefan Jaksic
E-mail: Stefan.Jaksic@ait.ac.at
Author(s): Abdelkader Shabaan (AIT), Siddhartha Verma (AIT), Wolfgang Herzner (AIT), Stefan Jaksic (AIT), Martin Matschnig (Siemens), Lukas Krammer (Siemens)
Doc ID: D3.2

Amendment History

Version	Date	Author	Description/Comments
V0.1	20.09.2019	S. Jaksic	Initial version prepared
V0.2	26.09.2019	S. Jaksic	SHSA
V0.3	07.10.2019	A. Shabaan	ThreatGet example introduced
V0.4	15.10.2019	S.Verma, A.Shabaan	Security tools
V0.5	25.10.2019	S. Jaksic	Corrections
V0.6	07.11.2019	S. Jaksic, W. Herzner	Appendix
V0.7	12.11.2019	S.Chlup, W. Herzner	GSFlow, Appendix, Executive summary
V0.8	15.11.2019	S.Chlup	Corrections and references
V1.0	19.11.2019	L.Krammer	Recommender System
V1.1.	25.11.2019	S. Jaksic	Post-review updates

The IoT4CPS project is partially funded by the "ICT of the Future" Program of the FFG and the BMVIT.

Federal Ministry
Republic of Austria
Transport, Innovation
and Technology

FFG
Forschung wirkt.

Content

Abbreviations

API	Application Programming Interface
AT	Attack Tree
ATA	Attack Tree Analysis
BAS	Basic attack steps
CR	Component Requirement
CPS	Cyber-Physical System
CPU	Central Processing Unit
DFD	Data Flow Diagram
DI	Data Input
DO	Data Output
DoS	Denial of Service
DNF	Disjunctive Normal Form
EA	Enterprise Architect
EDR	Embedded Device Requirements
FMEA	Failure Mode and Effects Analysis
FMVEA	Failure Mode, Vulnerabilities and Effects Analysis
FPGA	Field Programmable Gate Array
FTA	Fault Tree Analysis
HAS	Higher attack states
I4.0	Industry 4.0
IoT	Internet of Things
IIoT	Industrial IoT
KP	Knowledge Pack
MORETO	Model-based Security Requirement Management Tool
NDR	Network Device Requirements
OSF	Open Semantic Framework
QoS	Quality of Service
ROTS	Real-Time Operating System
SA	Security Achieved
SCPN	Stochastic Colored Petri Net
ST	Security Target
SHSA	Self-Healing through Structural Adaptation
SysML	System Modelling Language
TARA	Threat Analysis and Risk Assessment
TOE	Target of Evaluation
TS	Target System
UC	Use Case
UML	Unified Modelling Language
VBA	Visual Basic
V&V	Verification and Validation
WCET	Worst Case Execution Time
WP	Work Package
XML	eXtensible Markup Language

Executive Summary

In this deliverable we provide guidelines, processes and recommendations to build dependable IoT systems. D3.2 methods and tools tackle challenges along different CPS architecture layers: information layer, control layer and network layer. In this deliverable we also position WP3 contributions w.r.t. IoT4CPS use cases: Automated Driving and Industry4.0.

The physical-level tools and methods such as sensor security measures for discovering faulty and hacked sensors as well as will be reported in deliverables D3.4: “System architecture patterns for enabling multi-stakeholder trust provisioning during production and maintenance”. In addition, we report on forward-secure key exchange mechanism in our deliverable D3.6.1: “Prototype cryptographic library implementation”. On a network level we report on recommender systems to develop dependable IoT system, which can help users who want to build large IoT systems to choose the appropriate protocols and system configurations. Another method to achieve dependability, applied on a platform-level, is the Self-Healing by Structural Adaptation which allows systems to leverage implicit redundancy to achieve resiliency to failures. Solutions for trusted localization and orientation can be found in D3.4.

On an application level we report on tools for a variety of tasks in cyber security. We present ThreatGet, a tool that identifies, detects, and understands potential security threats in the foundation level of system models. Moreto is a tool for security requirements analysis and management using modelling languages such as SysML/UML. Our next contribution is a tool for standard-based product development management: GSFlow. It is one of the results of a more general effort to develop tools to support model-based development approaches and Safety & Security by Design. Last but not the least, we report on methods for safety and security risk assessment and formalize it into a verification pattern.

1. Introduction

Deliverable *D3.2: Guidelines, processes and recommendations for the design of dependable IoT Systems* is the second deliverable related to task *T3.1: Dependability design methods for IoT* and is strongly related to the deliverable *D3.1: Design and Methods Concept*. In D3.2 we further elaborate methods from D3.1 and explain how user can leverage those methods to obtain dependable systems. We also upgrade the deliverable with new methods and tools such as ThreatGet, Safety and Security co-engineering methodology and Verification and Validation (V&V) patterns.

For the sake of completeness, it is important at this point to provide the reference to the structure work w.r.t. architectural layers. Similarly to D3.1, in this deliverable we also collect and present potential solutions, tools and methodological building blocks for the development of safe and secure Internet of Things (IoT) and Cyber-Physical Systems (CPS). This deliverable puts a special focus on the integration of privacy and on the support of the complete engineering cycle, from engineering support to providing potential solutions.

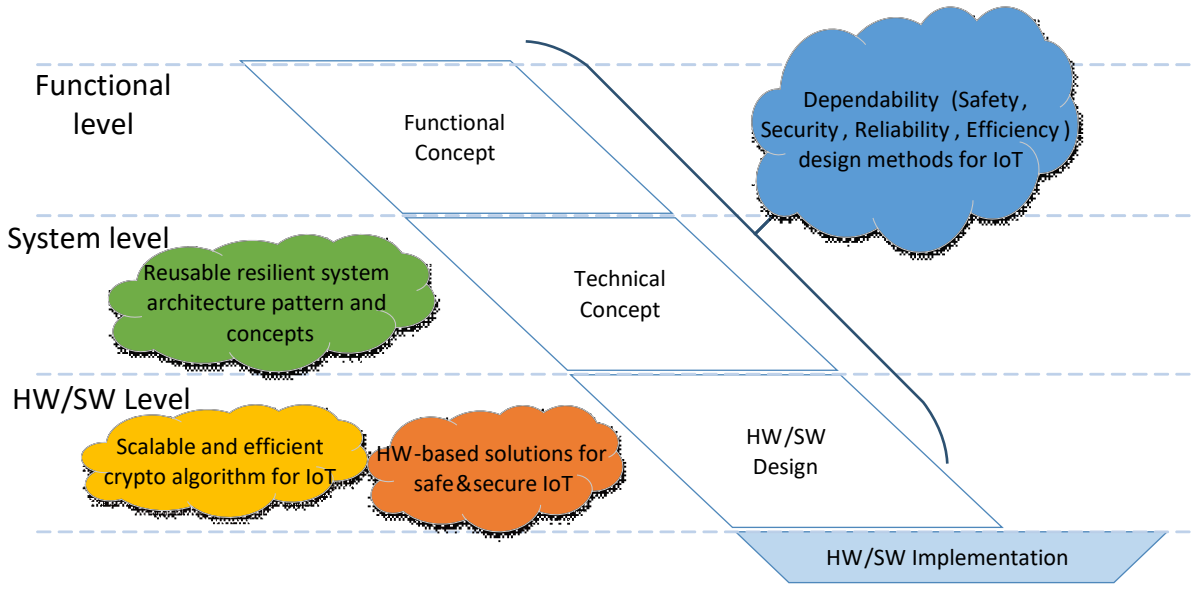


Figure 1: Structuring WP3 contributions along the V-Model

The contributions are structured based around the left side of the V-Model. The usage of the V-Model is here mainly as a structuring model and for easier classification and explanation. The usage of the V-Model should not be understood as an enforcement of this process model and the presented methods and building blocks are not restricted in term of engineering process model.

Our understanding of dependability itself is based on [39]. In this work, the dependability is differentiated between attributes, threats and means. The attributes summarize all parts of dependability, e.g. the set of properties we would like to ensure that a system we need to rely on possess. Due to the need of our industrial partners as well as the nature of our two main use cases, the IoT4CPS project sets priority on safety and security.

Threats are potential factors which can cause a violation or contribute to a violation of a dependability attribute. In order to protect the attributes, we can use different means. The following section will give a short overview about the basic system concept considered in IoT4CPS and present then different means to achieve dependability, which are considered in WP3 of IoT4CPS project.

Our tools, methods and guidelines are mostly non directly bound to a specific use case and can be applied to either of our IoT4CPS main use cases: Industry 4.0 as well as Autonomous Driving. Two of them which are tightly

coupled to the use case are the Autonomous Driving Platform developed by TTTech, as well as the Recommender System for IoT, developed by Siemens, for our Industry 4.0 use-case. The overview can be seen in Figure 2.

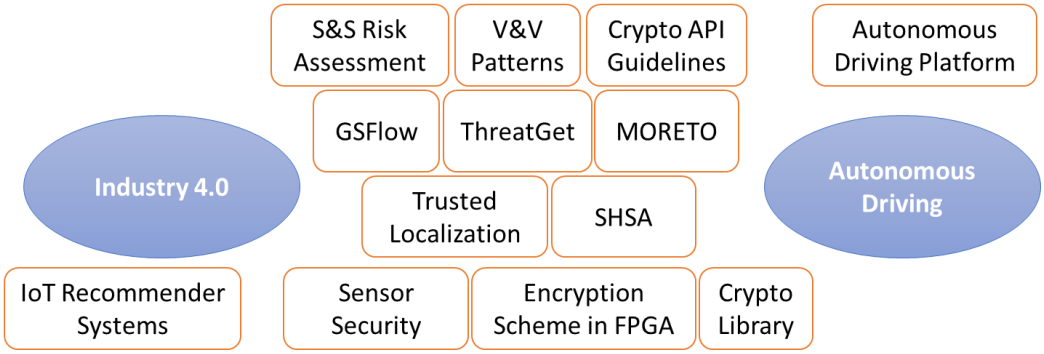


Figure 2: Most contributions of WP3 fits into both of IoT4CPS use-cases

2. Overview and System Model

In Figure 3 we can see how different WP3 contributions are positioned in the CPS Layer stack. A typical CPS is a complex system of systems which interact with one another. Depending on their purpose the components are usually classified into one of four layers: information layer, control layer, network layer and physical layer. The physical level is the lowest level in hierarchy. This is where data is obtained from sensors and data is prepared to be sent to other systems (i.e. encrypted). Once the system obtained the data, it is accessed and transmitted to interested components on network nodes. Typically, this takes place on network layer. On top of network layer, we see the platform layer. It is a layer where most system core functions are implemented, which are to be used by application layer.

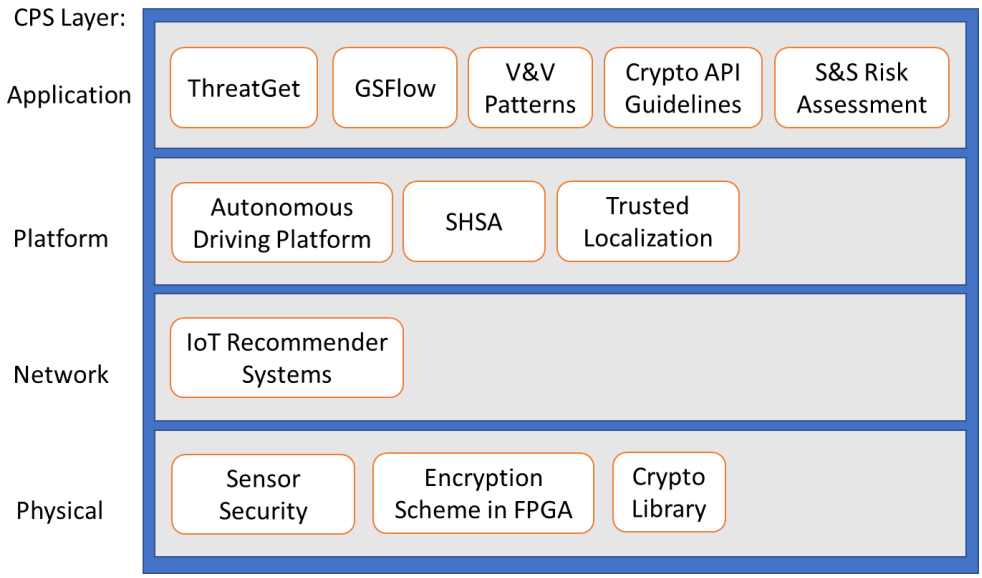


Figure 3: Contributions to safety and security can be separated into different CPS layers.

IoT4CPS project achievements can be found in any of the CPS architecture layers. On application layer, we have our V&V patterns and cyber security tools such as GSFlow, Moreto and ThreatGet. In addition, the guidelines for developing usable cryptographic APIs by SBA research belongs to the CPS application layer. Finally, we report on hybrid methods for safety and security risk assessment.

Underneath the application layer, in the platform layer we find Self-Healing for Structural Adaptation (SHSA) a technique for building resilient CPS which is developed by TU Wien. Solutions for trusted localization and orientation in space, useful in our Industry 4.0 use case are developed by TU Graz together with JKU Linz. Finally, we have an autonomous driving platform which is developed by TTTech. To build a reliable and functional IoT ecosystem one can use a recommender system, developed by Siemens. Last but not the least, AIT and TUG developed low-overhead encryption scheme, which is also implemented in FPGA by Siemens. DUK will provide concepts for achieving sensor security and low-level data integrity.

IoT4CPS tools should support dependability in several levels of CPS hierarchy. Our encryption solutions would enable efficient and secure communication channel between devices with limited resources, which is the case in a typical IoT scenario. In extreme cases of devices uncapable of encryption, we can at least guarantee data authenticity by applying digital watermarking techniques. IoT recommender system further enhances dependability by guiding the user to select the appropriate protocols, thus making the entire system more reliable. On a platform level, we increase the trust in system by leveraging methods for trusted orientation and localization. A system which adopts our SHSA techniques gains the ability to manage unpredictable component failures, thus becoming more dependable. Finally, our design-time security and safety tools, V&V patterns and Cryptographic API development guidelines aim to increase dependability early on.

3. Application Layer Tools and Methods

In this section we focus on IoT4CPS WP3 contributions for improving dependability on the application level. We report on our design time security analysis tools for model-based devilment, for security requirement management and threat modelling as well as safety and security risk assessment.

3.1 GSFlow

Our first contribution is a tool for standard-based product development management: GSFlow. It is one of the results of a more general effort to develop tools to support model-based development approaches and safety & security by design.

The goal of GSFlow is to support the complete engineering lifecycle of safety and/or security relevant systems based on pre-defined processes, by guiding the user through the development process. Its main objective is to make standard driven development straightforward, especially for companies that are unacquainted with functional safety and security standards. The model implemented in GSFlow simplifies standard driven development by guiding the end-users through the development process and consequently offloads the effort from security experts, while still providing assurance. This is especially relevant to SMEs which only have a limited number of safety or security experts.

The central output of GSFlow is a safety and/or security case which shall support companies two ways: (1) helping to reach assurance for their products and (2) allowing to summarize the argumentation why a system is acceptably safe and secure. To achieve this goal every project in GSFlow contains requirements which correspond to functional safety and security standards such as [37]. GSFlow provides standard conformant user management and utilizes tools to ensure the quality of an evidence.

GSFlow provides a flexible framework for modelling and executing standards. It is also capable of executing plugins written by an external developer. This flexibility ensures that the specific needs of a company can be met. Furthermore, an external developer is only required to implement an interface according to their needs. For example, GSFlow can execute external plugins to generate reports, as well as to check the artefacts and requirements using Natural Language Processing methods.

3.1.1 GSFlow Structure and Definitions

In this section we provide more details about the operation of GSFlow. Requirements are defined as the entities needed to achieve the objective of the project. Two different kinds of requirements can be distinguished. The first kind of requirements are the standard requirements, which are derived from functional safety and security standards. They are needed to identify the goals that are obligatory to reach compliance to relevant standards. Secondly, the product requirements in GSFlow represent the requirements that are specific to a product. They can be linked to a standard requirement. Furthermore, the requirements in GSFlow are the key elements for documenting the workflow as they serve as an anchor to attach evidence and trace every action that is conducted on them while being processed. Once processing is completed and all evidence has been created, a requirement may enter the state of completion.

Phases in GSFlow represent phases of a safety/security standard. GSFlow ensures that in every phase Standard Requirements that are specific to this phase need to be fulfilled. A phase can only start once all previous phases have been completed. Phases in GSFlow are used to structure requirements and define the order of execution in the workflow.

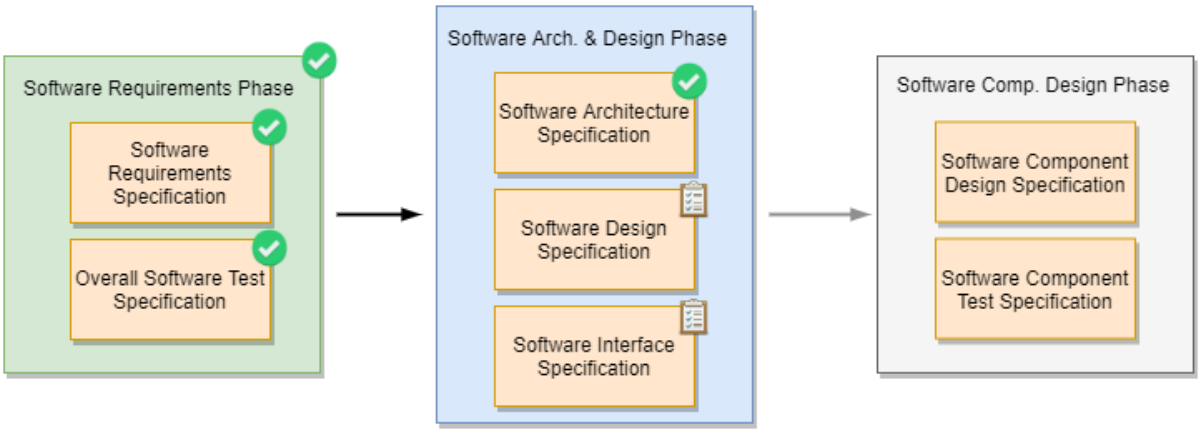


Figure 4: Illustration of workflow and fulfilment status

GSFlow allow for integration with external tools. By implementing and using the interfaces provided by GSFlow, an external Developer can create plugins and an admin can upload them to GSFlow after conducting validation. These tools/plugins are then ready to be executed. The dataflow between GSFlow and the tool/plugin is conducted only through the API. The utilization of tools shall serve as quality assurance as well as provide convenience to the users working with GSFlow.

GSFlow supports standard conformant user management. Different roles can be assigned to different users per project. Roles are based on a generic model of roles defined in different standards. In GSFlow, roles can be mapped to specific standards. The list of the supported roles includes:

- Project Manager
- Requirements Manager
- Developer
- Verifier
- Validator
- Assessor

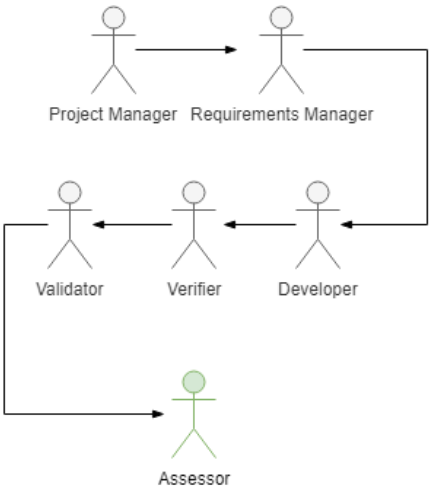


Figure 5: Basic responsibility chain in GSFlow

In GSFlow responsibilites are modelled on a per requirement basis. Each requirement has the required roles assigned to support the development chain defined in a standard. To be more precise, when the developer has finished their task, the verifier is assigned followed by the validator. Only when all the the responsible parties have marked their tasks as finished, assessment can take place.

As discussed, GSFlow enables the end user to organize their workflow according to phases, work products and requirements deducted from standards. GSFlow also serves as a documentation platform and tracks every action that is conducted regarding a certain requirement and links evidences to those requirements. This way, GSFlow supports traceability. The flexible framework inside GSFlow enables tailoring of generic processes to company or project specific demands. When appropriately modelled, it can support a safety and security co-engineering workflow. To achieve this, different adequate standards need to be analysed and consequently modelled into one combined workflow.

For example, the standards ISO/IEC 27002[35] and EN 50128 [34] contain requirements and methods emphasizing on availability, reliability, confidentiality, integrity and maintainability. These standards describe measures that need to be undertaken in order to assure the aforementioned attributes. SAE Standard J3061 [37], which is a Cybersecurity Guidebook for Cyber-Physical Vehicle Systems, describes security and its effects on safety on financial and operational basis and hence, availability and reliability. Security and measures to ensure integrity and confidentiality can be found in the IEC 62443 series [38]. By following the generated workflow, GSFlow makes the development of dependable IoT systems feasible.

3.2 Failure Mode, Vulnerabilities and Effects Analysis

Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) is a static method for security analysis. FMVEA is based on the Failure Mode and Effect Analysis (FMEA) and extends the standard approach with security related threat modes [23]. A *failure mode* describes the way the function of an element fails. A *threat mode* describes the way in which the identified function of an element can be misused. Threat modes classifies threats in six categories (Spoofing of user identity, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege).

FMVEA consists of several phases, beginning with system modelling phase. Once this phase is complete the failure and threat modes for each element of the system model are identified. Depending on the domain, the system architecture and the knowledge about the system, failure and threat modes can be refined and extended. Each identified failure or threat mode associated with the element is investigated for potential effects. For modes with critical effects, potential causes are analysed and the likelihood for each cause is estimated. For threat modes, likelihood is determined using a combination of threat and system properties.

Threat properties mainly describe the resource and motivation of a potential threat agent while system properties include reachability and system architecture. The system model is based on a three-level data flow diagram (DFD). Effects of failure and threat modes are presented at the context level of the diagram, which shows the interaction between the system and its environment. Failure and threat modes are located at the level 1 DFD. Vulnerabilities and failure causes are based on the level 2 DFDs.

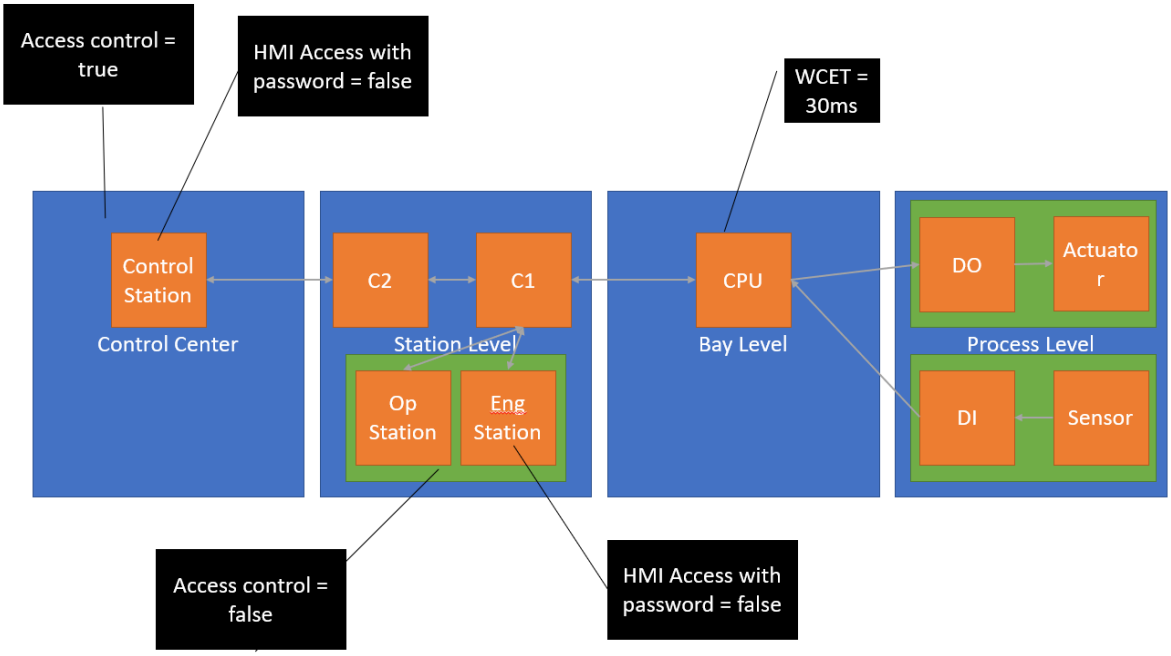


Figure 6: An example model as an input for analysis with FMVEA

Figure 6 shows the diagram of an example use-case in which a set of actors is controlled by an CPU, based on some sensor input. Control Strategy can be defined and monitored in some higher layers. The example model will be used to perform an analysis with FMVEA. The blue squares represent the root environments of a specific level. Each orange node can be seen as an operating element inside the environment. Green squares represent sub-environments inside the root-environments which encapsulate their own operating elements. Black squares display the defined attributes/properties for the diagram elements. For example, the “CPU” has the attribute/property “WCET” (“Worst Execution Time”) with a value of “30 ms”.

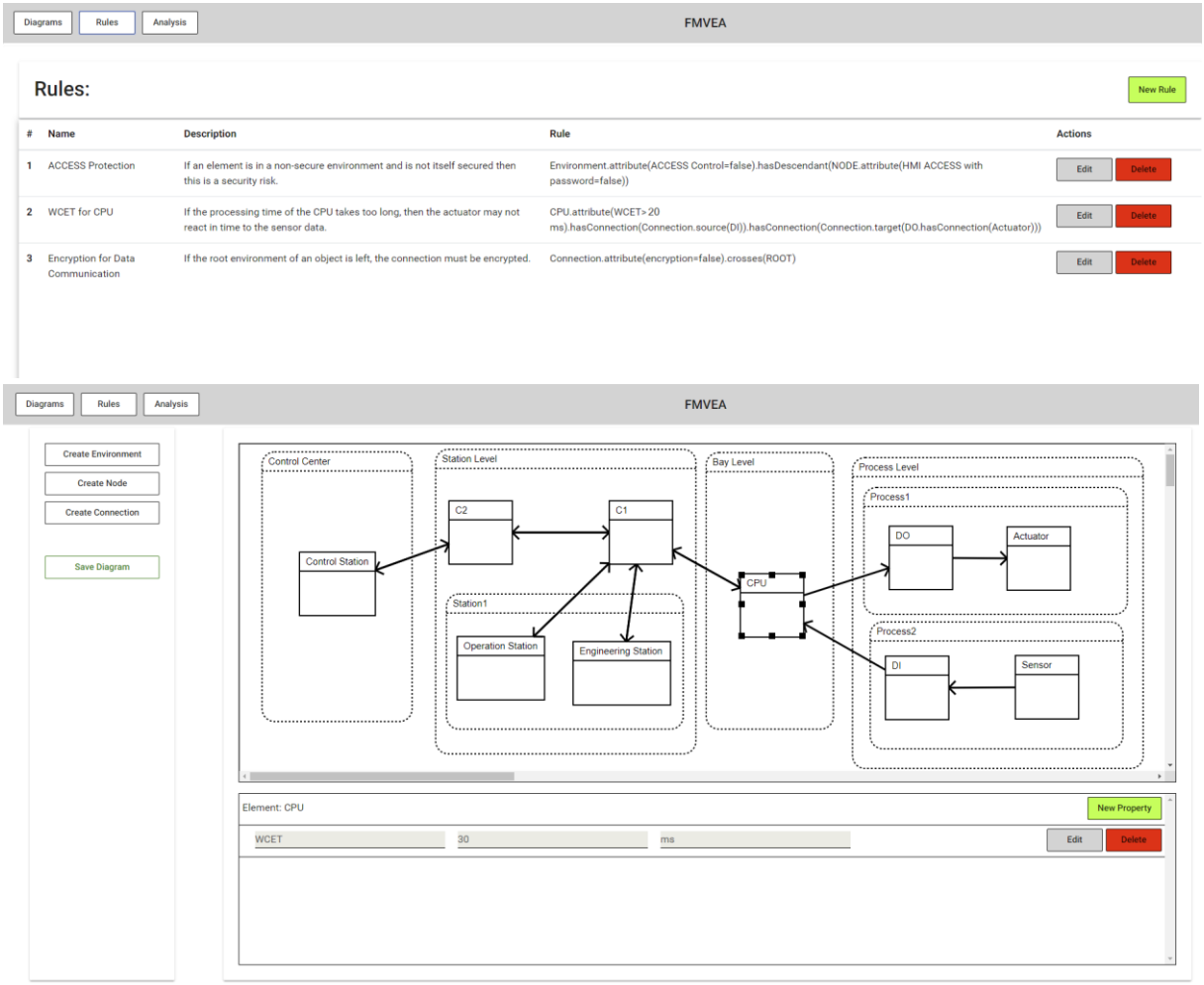


Figure 7: The diagram modelled in FMVEA and the rules

Figure 7 shows the diagram from Figure 6 modelled in FMVEA. On the left-hand side, we can see the diagram related actions like “Create Environment”, “Create Node” and “Create Node”. An Environment can be considered as a container, which provides general attributes to his children. Attributes can be focused on Security and Safety. The attributes of an element are directly displayed below the diagram. Figure 7 also displays the rules which should be used to analyse the use-case diagram. From the left to the right are the names of the rule then a short description and the “Rule”. The “Rule”-column is the most important one, because here the actual rule for the analyser is defined. The content of a rule is defined by the grammar shown in Figure 8.

```
Expression → ( ObjExpression | ConnectionExpression )
ObjExpression → ( SingleObjExpression | MultipleObjExpression )  AttributeFilter:? ConnectionFilter:? RelationFilter:*
SingleObjExpression → Identifier
Identifier → singleString
MultipleObjExpression → "{"  ( AndConnectedObjects | OrConnectedObjects ) "}"
AndConnectedObjects → "{" ObjExpression _ "AND" _  ObjExpression  ( _ "AND" _ ObjExpression ):* "}"
OrConnectedObjects → "{" ObjExpression _ "OR" _  ObjExpression  ( _ "OR" _ ObjExpression ):* "}"
AttributeFilter → ".attributes{" Attribute "}"
Attribute → (SingleAttribute | MultipleAttribute)
SingleAttribute → ( StringValueAttribute | NumericValueAttribute | MultipleStringValueAttribute| MultipleNumericValueAttribute)
MultipleStringValueAttribute → multipleString "IN {" singleString:+ "}"
MultipleNumericValueAttribute → multipleString "IN {" ((singleNumber | decimalNumber) (_ attributeUnit):?):+ "}"
StringValueAttribute → multipleString "=" singleString
NumericValueAttribute → multipleString ("=" | ">" | "<" ) (singleNumber | decimalNumber) (_ attributeUnit):?
MultipleAttribute → (AndConnectedAttributes | ORConnectedAttributes
AndConnectedAttributes → Attribute ("," _ Attribute):+
ORConnectedAttributes →  "{" Attribute ( _ "OR" _ Attribute ):+ "}"
ConnectionFilter → ".hasConnection{Connection" (SourceFilter | TargetFilter) CrossesFilter:? AttributeFilter:? "}"
SourceFilter → ".from{" ObjExpression "}"
TargetFilter → ".to{" ObjExpression "}"
CrossesFilter → ".crosses{" ( "PARENT" | "ROOT" ) "}"
RelationFilter → ( ".hasDescendant{" ObjExpression "}" | ".hasAncestor{" ObjExpression "}" )
ConnectionExpression → "Connection" SourceFilter:? TargetFilter:? CrossesFilter:? AttributeFilter:?
singleString → [a-zA-z]:+
multipleString → [a-zA-z]:+ ( _ [a-zA-z]:+ ):*
singleNumber → [0-9]:+
decimalNumber → [0-9]:+ "." [0-9]:+
attributeUnit → [a-zA-z/]:+
```

Figure 8: Example of the FMVEA grammar

Figure 9 displays the results of the use-case analysis. The previously created rules are applied on the selected diagram.

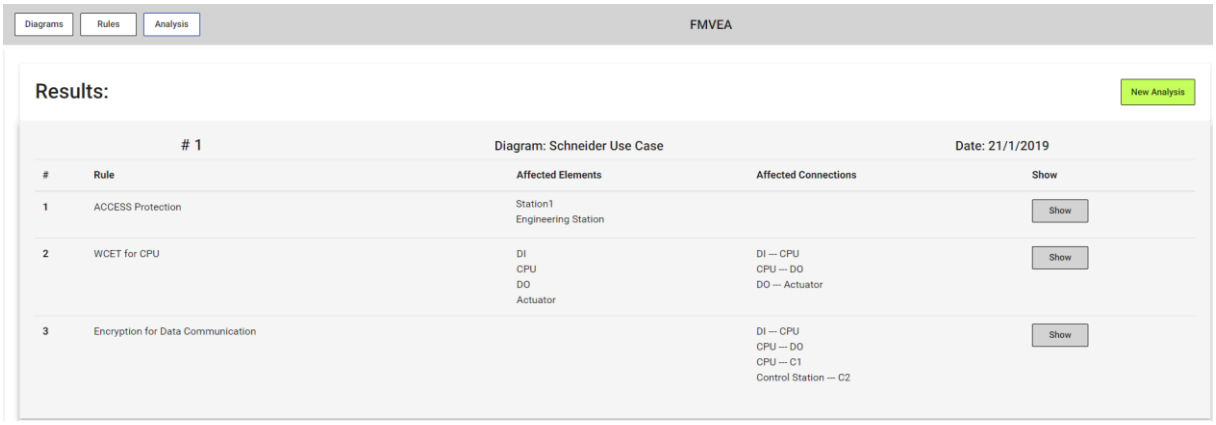


Figure 9: Analysis results for the defined rules and the diagram

From left to right you one can observe the applied rule and the results of the specific rule on the diagram. The affected elements and connections can be viewed in the diagram if the user clicks on the “Show”-button to the right. Inside the diagram the affected elements and connections get highlighted by a red border as you can see in the Figures 11-13.

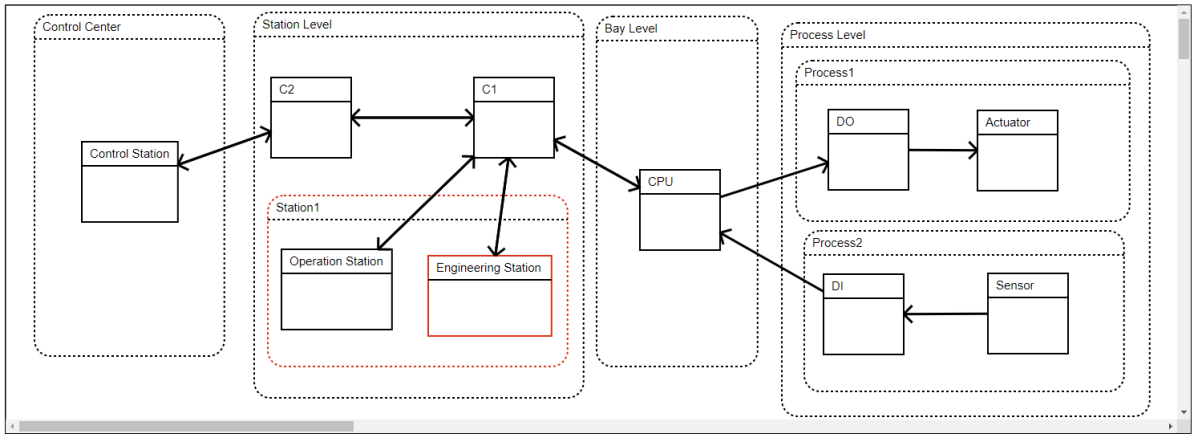


Figure 10: 1st rule results

In Figure 10 you can see the affected elements of the first rule. The Definition of the first rule says that if there is any environment with the property “ACCESS Control=false” which contains a child object with the property “HMI ACCESS with password=false” then there is a security problem. As one can observe from Figure 6 we initially defined the “Control Station” with “ACCESS Control=true” so there is no problem even if the “Control Station” has the property “HMI ACCESS with password=false”. However, if one observes the “Station 1” and the “Engineering Station” then both criteria are fulfilled, and this is the reason why these two objects are the affected objects of the first rule.

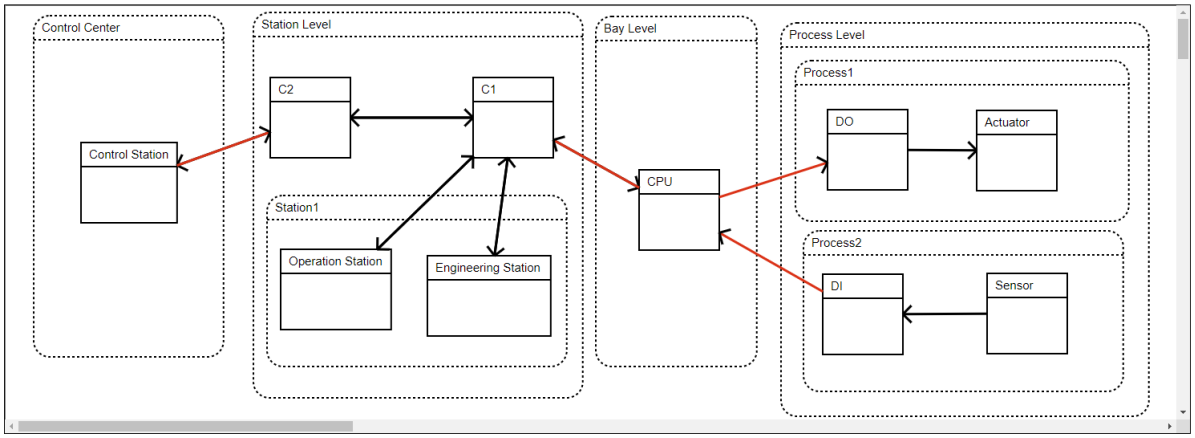


Figure 11: 2nd rule results

In Figure 11 you can see the affected connections of the second rule. The Definition of the second rule says that if there is any connection between two objects which do not share the same root object then the connection must be encrypted. One can observe from Figure 6 we never defined an encryption property for any connection inside the diagram. For example, take the connection between the “Control Station” and “C2”. The root object of the “Control Station” is the “Control Center” and the root object of the “C2” is the “Station Level” but they share a connection. Now let’s consider the connection between the “C2” and “C1”. The root object of both objects is the “Station Level”. They share the same root element as the same parent element, therefore this connection does not represent a potential problem.

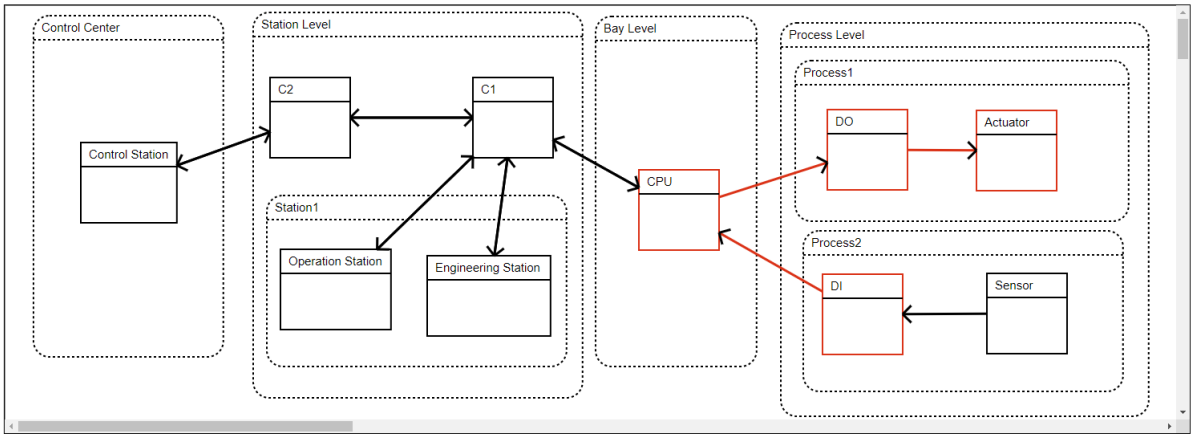


Figure 12: 3rd rule results

In Figure 12 you can see the affected elements and connections of the third rule. As you can take from Figure 6 we set the WCET of the CPU as “30 ms”. The Definition of the third rule says that if the WCET inside the CPU is higher than “20 ms” the actuator may not react in time to the sensor data. The sensor data is coming from the “Data Input” (DI) then the data is processed inside the CPU with a WCET of 30ms and then sent to the “Actuator” over the “Data Output” (DO). This is the reason why the “DI”, “CPU”, “DO” and the “Actuator” are affected elements in this case. The connections between these objects transport the data and are affected connections in this case. This rule could be expanded by additionally taking into account the latency of the connections.

3.3 Threat Modelling

A secure system can be designed and developed only if security issues are well-identified and addressed appropriately in the early stages of the system development. That is considered a significant advantage because once the system is developed, it becomes harder to add security countermeasures. ThreatGet is a toolbox for Enterprise Architect, which is a widely used tool for Model-Based Systems Engineering. ThreatGet identifies, detects, and understands potential threats in the foundation level of system models. It supports the initial steps of the developing system process to guarantee the security by design. The following figure depicts an example design which a user can specify in ThreatGet, in order to perform security threat analysis.

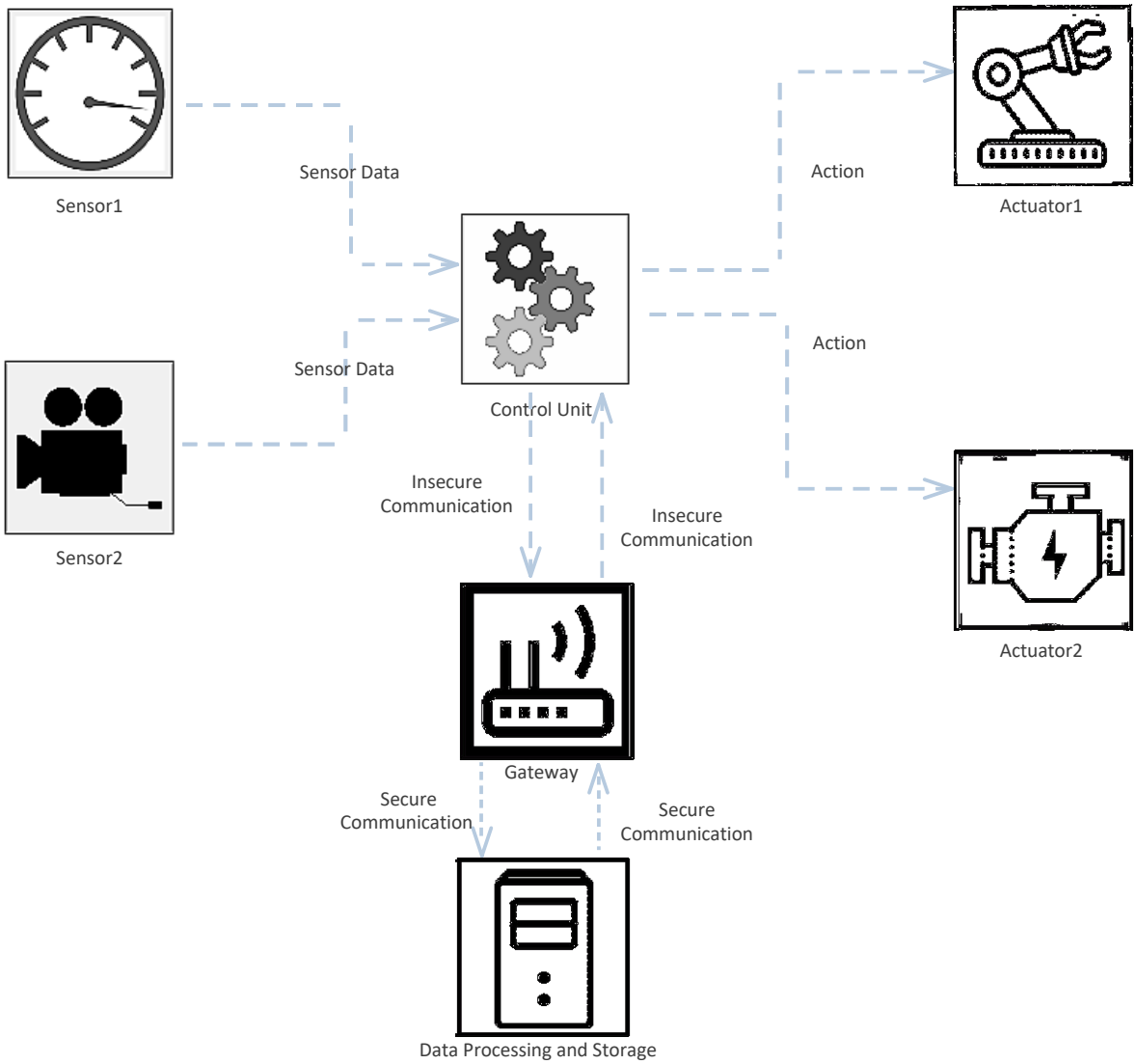


Figure 13: IoT-based Smart Factory

Our Figure 13: IoT-based Smart Factory illustrates an example of IoT application in a smart factory, which is one of two main use cases of IoT4CPS project. The example contains one or more sensors and camera units for collecting and gathering information about the production line. The collected data is processed by a control unit, that handle and manage actions by sending signals to actuator units such as robotic arm and engine as defined in the figure. The data is sent to a centralized data storage and processing unit for monitoring the quality of the production.

In such a heterogeneous and distributed system, potential security threats can exist in any component, which may compromise the operation of the entire system. In order to identify potential threats in our system, we apply ThreatGet. The threats are defined according to the dataflow from the source components to the targets. According to the security properties of these units, some vulnerabilities could be exploited be threats.

In Figure 13 the Control Unit takes the central role in the Smart Factory model. This component communicates with the data storage through a gateway, which runs a certain communication protocol. Depending on the gateway device, it can provide low or high security features. In our model, we can analyze devices with different levels of security by adjusting the possible security parameters of the model. We model possible security mitigation measures in the communication flow between the Control Unit and the Gateway. The features include: Source and Destination Authentication, Confidentiality and Integrity. We can set these parameters to values which correspond to a real device.

In Figure 14 we can observe that if the HTTP protocol has mitigation measures switched off, the communication channel introduces 6 different threats. Once we introduce the mitigation measures in the communication channel the number of threats reduces to 4.

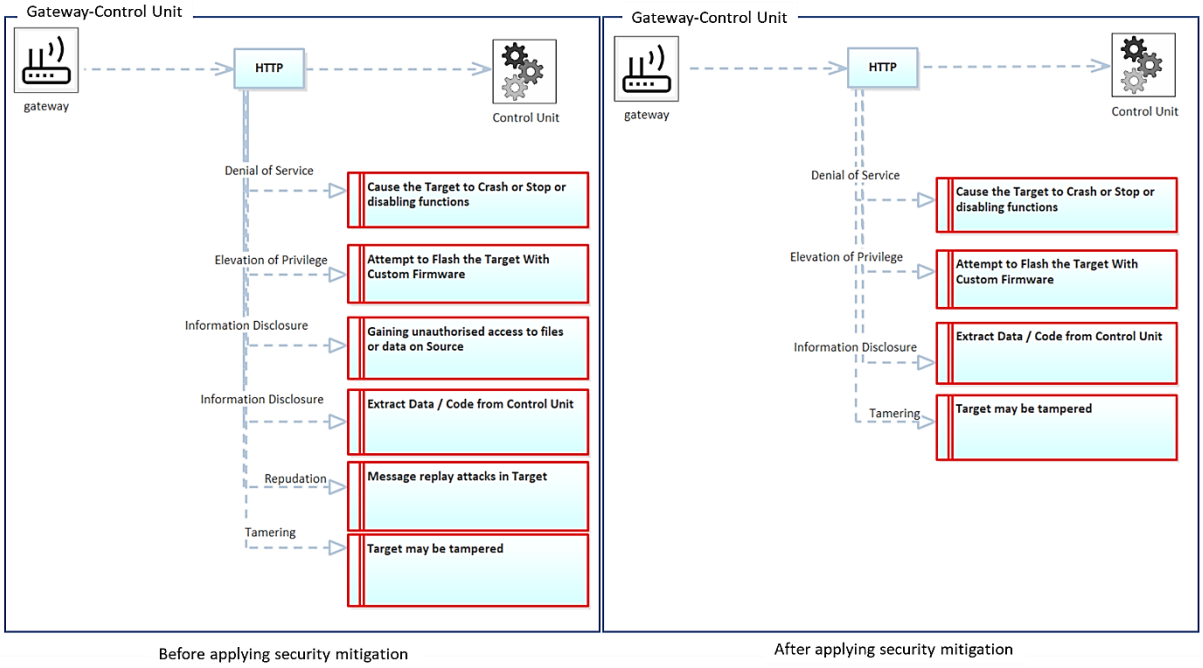
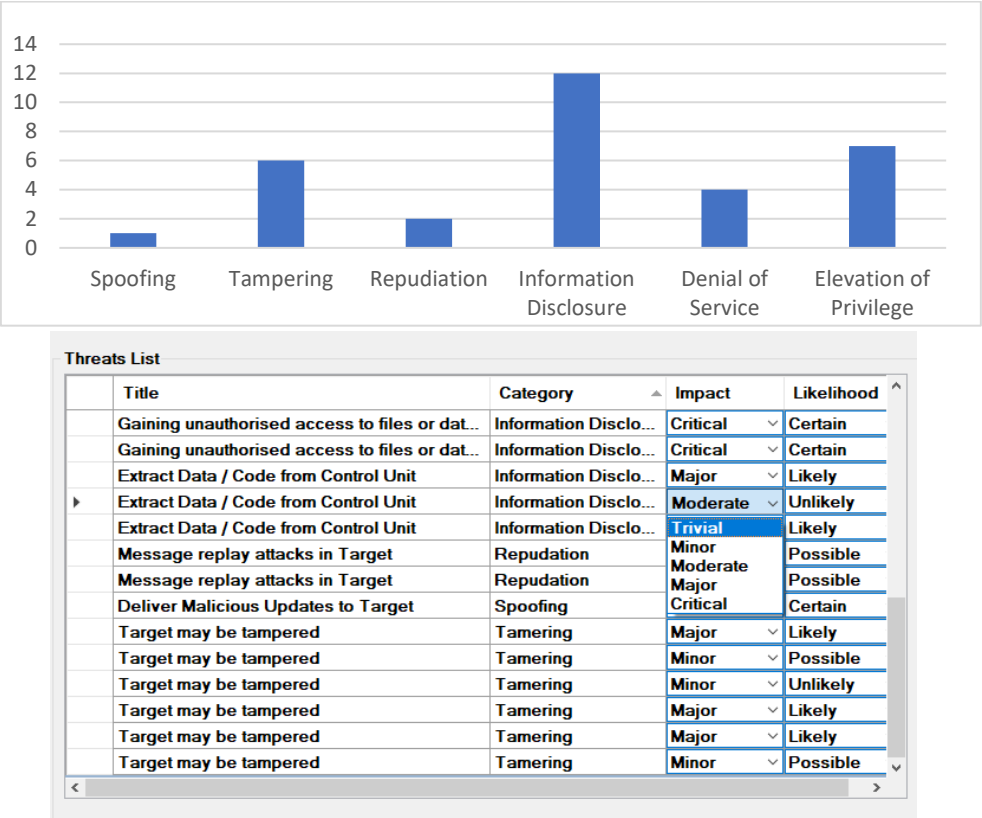


Figure 14: The impact of introducing a mitigation measure

As a result of the analysis, ThreatGet detects 32 potential threats, that are classified according to the STRIDE model (i.e., Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS), and Elevation of Privilege). Spoofing represents an attempt by a person or program to identify itself as another by falsifying data, to gain an illegitimate advantage. Data tampering is an attempt to maliciously modify the data through unauthorized channels. Repudiation is a kind of attack which manipulates the log data in the computer systems, in order to conceal traces in the log. Denial of Service and Elevation of privilege are well-studied threats, where an attacker is jamming the access to the system resource, and when an attacker attempts to gain more access rights than allowed, respectively. The number of the detected threats according to the STRIDE model is depicted in Figure 15.



Threats List

Title	Category	Impact	Likelihood
Gaining unauthorised access to files or dat...	Information Disclo...	Critical	Certain
Gaining unauthorised access to files or dat...	Information Disclo...	Critical	Certain
Extract Data / Code from Control Unit	Information Disclo...	Major	Likely
Extract Data / Code from Control Unit	Information Disclo...	Moderate	Unlikely
Extract Data / Code from Control Unit	Information Disclo...	Trivial	Likely
Message replay attacks in Target	Repudation	Minor Moderate Major Critical	Possible
Message replay attacks in Target	Repudation		Possible
Deliver Malicious Updates to Target	Spoofing		Certain
Target may be tampered	Tamering	Major	Likely
Target may be tampered	Tamering	Minor	Possible
Target may be tampered	Tamering	Minor	Unlikely
Target may be tampered	Tamering	Major	Likely
Target may be tampered	Tamering	Major	Likely
Target may be tampered	Tamering	Minor	Possible

Figure 15: Distribution of identified threats according to the STRIDE model and all potential threats listed

Afterwards, the risk assessment process is applied to evaluate risk severities of the detected threats according to the values of the impact and likelihood parameters. The ThreatGet's user can select the parameters of the impact and likelihood to estimate the risk severity, as shown in Figure 15. There are five parameter values of the Impact and Likelihood are used in the estimation process to determine the risk severity level of threats. Figure 16 illustrates the levels of the impact and likelihood that are used by ThreatGet for the risk assessment process.

		Likelihood				
		1 Remote	2 Unlikely	3 Possible	4 Likely	5 Certain
Impact	1 Trivial	1	2	3	4	5
	2 Minor	2	4	6	8	10
	3 Moderate	3	6	9	12	15
	4 Major	4	8	12	16	20
	5 Critical	5	10	15	20	25

*Risk = Threat * Vulnerability * Consequence*

Threat * Vulnerability = Likelihood

Consequence = Impact

Low 1:5 Medium 6:10 High 11:16 Extreme 17:25

Figure 16: Risk assessment chart

The next step is to address the identified potential threats by the most applicable security countermeasure for mitigating the overall risk of the predefined system model. Furthermore, we use IEC 62443 as the most suitable security standard to provide a flexible framework for addressing current and future security vulnerabilities could be exploited by potential threats [29]. In this case we should define two main values that could help to reach the high level of security protection. The first one is called the Security Achieved (SA), this value defines the current security level that is achieved after applying security countermeasures against security vulnerabilities. The second value is to define the actual security goal that the system architect wants to be achieved this parameter is called Security Target (SA) [28].

In this example we claim that we are looking forward to adding all the threats with the highest severity level. Furthermore, the SA = 7 (the total number of extreme potential threats in this example), and SA = 0 because there are no security countermeasures are applied yet. The following part discuss the security treatment process for addressing the security threats by security countermeasures.

3.3.1 Risk Treatment for the Identified Extreme Threats

This process plays an important role to address potential threats with security countermeasures that should be considered in the system developing phases to keep the risk always low. The system security target (ST) is defined to estimate the maximum-security level needs to be achieved. Also, the current status of the system security level is defined as the security achieved (SA).

In this example, we are looking forward to reducing the security level of the unacceptable risk level. Furthermore, we focus on the Extreme severity level of threats (unacceptable risk level) that need to be addressed by proper security countermeasure. Thuss, ST is seven (the total number of extreme severity risk), and SA is one until addressing these threats. The value of SA is incremental according to change in security level after applying additional security countermeasures. This process is completed if SA = ST; otherwise, additional security countermeasures should be applied for security improvement. Figure 17 illustrates the difference between the current and the target security levels before applying security countermeasure.

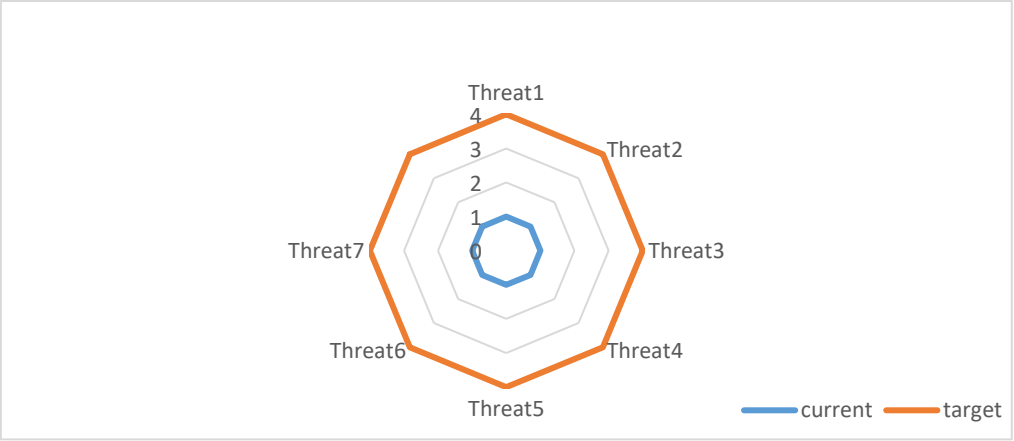


Figure 17 The current (blue) and target (orange) security levels before applying security countermeasures

We proceed with handling security issues threat by threat and showing the outcome of applying a mitigation measure. The following sections define the selected security countermeasure based on IEC 62443 part 4-2 for addressing the existing potential threats.

3.3.2 Risk Treatment for Threat 1 and Threat 2

Threat1 Title: Message replay attacks in Target

Threat1 Category: *Information Disclosure*
Threat1 Origin:

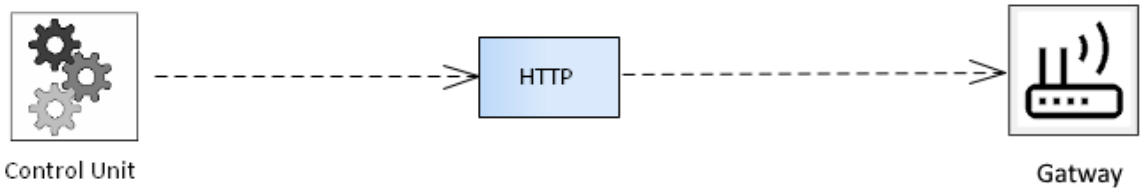


Figure 18 Threats 1 and 2 propagation source

The following table shows a list of selected security countermeasures for addressing this threat. These security countermeasures should be applied to mitigate the threat. The following abbreviations are used: CR: Component Requirement, NDR: Network Device Requirements, and EDR: Embedded Device Requirements.

IEC62443 Classification	Name of the security requirement
CR 2.11	Timestamps
CR 2.10	Response to audit 48 processing failures
CR 2.12	Non-repudiation
NDR 3.14	Integrity of the boot process
EDR 3.14	Integrity of the boot process

Table 1 IEC62443 Requirements for Threat1

Threat2 Title: Target may be tampered
Threat2 Category: *Tampering*
Threat2 Origin: *shown in Figure 18*

The following table defines a list of the chosen security countermeasures for addressing threat2.

IEC62443 Classification	Name of the security requirement
NDR 3.11	Physical tamper resistance and detection
CR 1.9	Strength of public key-based authentication
NDR 3.14	Integrity of the boot process
EDR 3.11	Physical tamper resistance and detection
EDR 3.14	Integrity of the boot process

Table 2 IEC62443 Requirements for Threats 3&4

3.3.3 Risk Treatment for Threat 3 and Threat 4

Threat Title: Gaining unauthorized access to files or data on Source
Threat Category: *Information Disclosure*
Threat Origin:

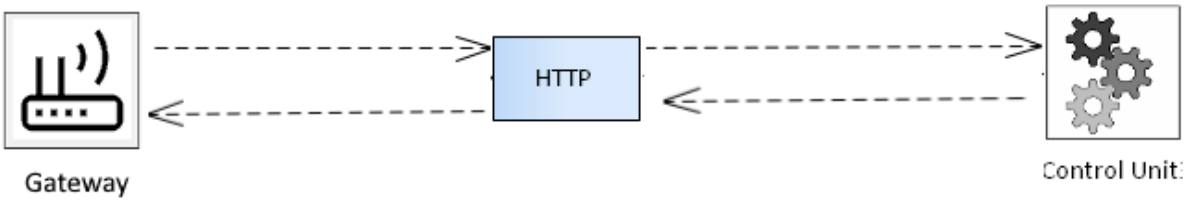


Figure 19 Threats 3 and 4 propagation source

The following table shows the chosen security requirements for these threats.

IEC62443 Classification	Name of the security requirement
CR 4.1	Information confidentiality
CR 4.3	Use of cryptography
CR 3.1	Communication integrity
EDR 3.2	Protection from malicious code
NDR 3.2	Protection from malicious code
CR 3.8	Session integrity

Table 2 IEC62443 Requirements for Threats 3&4

3.3.4 Risk Treatment for other Threats

Threat5 Title: **Services from back-end server disrupted**
Threat5 Category**:** Elevation of Privilege
Threat5 Origin**:** *shown in Figure 19*

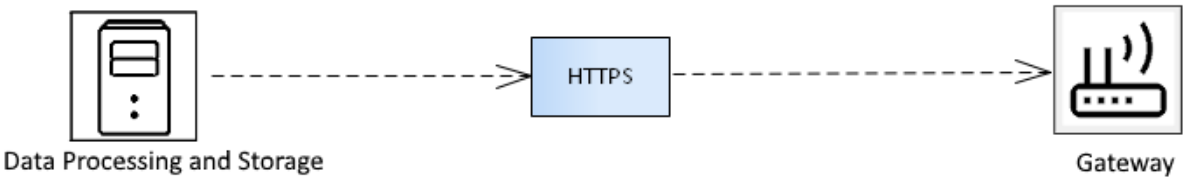


Figure 20 Threats 5, 6 and 7 propagation source

Table 4 shows the selected security countermeasures for threat5.

IEC62443 Classification	Name of the security requirement
CR 1.1	Human user identification and authentication
CR 2.1	Authorization enforcement
HDR 3.10	Support for updates
NDR 3.10	Support for updates
HDR 3.14	Integrity of the boot process
NDR 3.14	Integrity of the boot process

Table 3 IEC62443 Requirements for Threat 5

Threat6 Title*:* Deliver Malicious Updates to Target
Threat6 Category: *Spoofing*
Threat6 Origin: *shown in Figure 20*

Security Countermeasures:

The following Table lists the possible security countermeasure that should be applied to mitigate the risk of Threat6.

IEC62443 Classification	Name of the security requirement
CR 3.8	Session integrity
CR 1.2	Software process and device identification and authentication
CR 1.5	Authenticator management
HDR 3.2	Protection from malicious code
NDR 3.2	Protection from malicious code
CR 1.7	Strength of password-based authentication
CR 6.2	Continuous monitoring

Table 4 IEC62443 Requirements for Threat 6

Threat7 Title: Cause the Target to Crash or Stop or disabling functions

Threat7 Category: Denial of Service

Threat7 Origin: *shown in Figure 22*

Security Countermeasures:

The following list of security requirements are selected to address the potential threat7.

IEC62443 Classification	Name of the security requirement
CR 1.11	Unsuccessful login attempts
CR 7.1	Denial of service protection
NDR 5.2	Zone boundary protection

Table 5 IEC62443 Requirements for Threat 7

3.4 MORETO

The correct security requirement identification and efficient security requirement management are essential for any security engineering process. We can design, implement, and test a secure system only if we know the exact security requirements. Achieving an efficient requirement management is a challenge in system development. The Model-based Security Requirement Management Tool (MORETO) serves a tool for security requirements analysis, allocation, and management using modelling languages such as SysML/UML. MORETO is an Enterprise Architect (EA) plugin for managing the IEC 62443 security standard. It is a reliable and a flexible to model safety & security requirements suited to different components and system architectures. It generates a list of security requirements in a given diagram, which can help the user to build-up a secure infrastructure. Figure 21 shows a simple example of different components which interact together through a network.

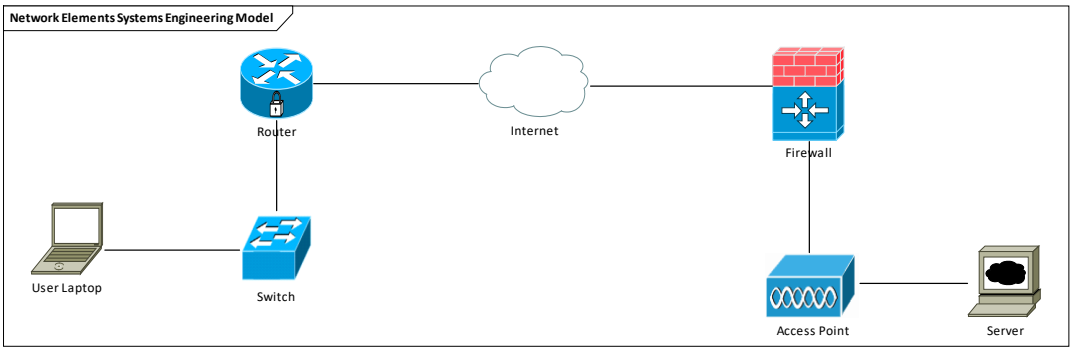


Figure 21 Simple network elements for system engineering model by MORETO

MORETO scans all the elements of a given model and automatically generates a list of security requirements based on expert knowledge encoded in the MORETO knowledge base itself and on the description contents of the IEC 62443. Figure 22 shows a list of identified security requirements of the Router and Switch devices respectively.

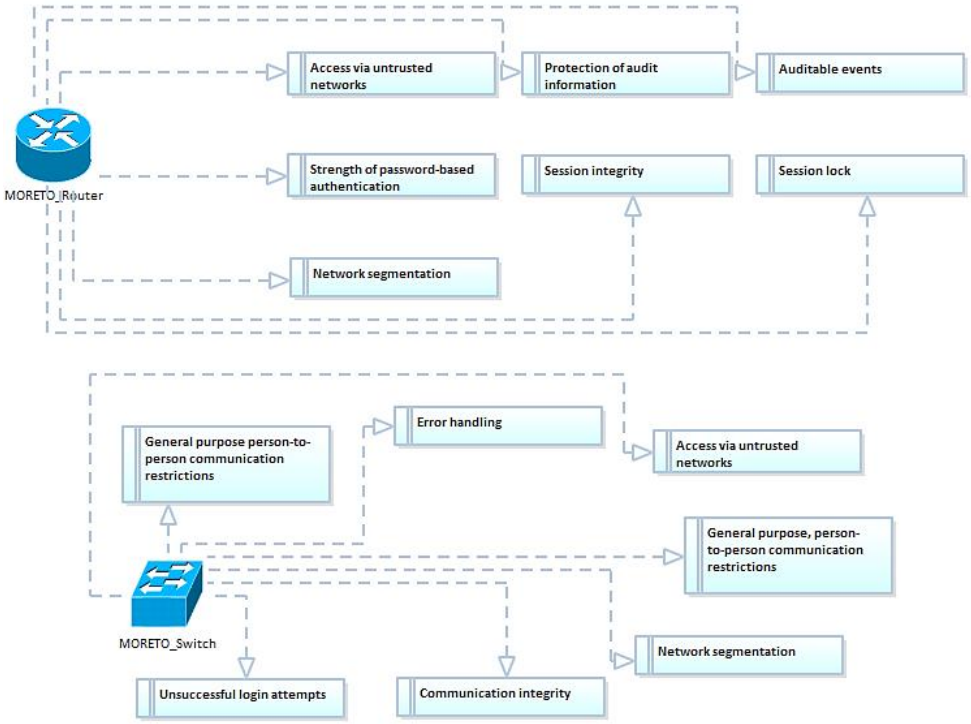


Figure 22 IEC 62443 Security standards for router and switch devices

3.5 Safety and Security co-engineering

In order to build dependable IoT systems, it is insufficient to regard concepts of safety and security separately since safety and security of IoT are not independent. Therefore, we propose a combined risk assessment approach for safety and security and demonstrate it using tools developed in IoT4CPS project.

The state-of-the-art approaches in Threat Analysis and Risk Assessment (TARA) such as Microsoft STRIDE and FMVEA provide a risk assessment approach but they do not consider multiple attacks (attack combinations via 'OR', 'AND', 'SAND' (sequential AND)) and multi-stage attacks. On the other hand, Attack Tree Analysis (ATA) approach considers such attacks but either in a purely qualitative or purely quantitative fashion (which is not feasible for most applications). There is no TARA-based approach which considers how the likelihood (final risk) can be evaluated in case of multiple attack combinations and multi-stage attacks.

The security risk assessment is performed combined with ADTool [30], which provides a simple way to create and edit attack trees. Attack trees are conceptual diagrams showing how an asset, or target, might be attacked. Attack trees have been used in a variety of applications. The attack tree is then exported as an XML file to the scripts [32] which implement the risk assessment algorithm. During evaluation two inputs are required by the user: the threat level scores for all the BAS (Basic Attack Steps) and Impact level scores for all the Higher Attack States.

3.5.1 Security Risk Assessment with Attack Trees

The generation of attack trees, which is a prerequisite for Attack Tree Analysis, is presently not covered in our work and is a subject of future work. The approaches such as Microsoft STRIDE generate general threats corresponding to STRIDE category by a predefined mapping of these threats to general elements of a system (approach represents almost all systems with four general elements). Therefore, the STRIDE and Data Flow Diagram (DFD) approach are sufficient to generate threats for each element. However, to generate an attack tree we also need to know how these threats affect each other, which can be learned from security attack databases. In case of automotive, AUTO-ISAC (SAE Vehicle Electrical System Security Committee) or public Automotive Attack Database [31] can be considered as a relevant data source. The concrete specification of system design is not known in the early stages of design process, which renders the very large list of all the possible attacks, thus making the STRIDE approach preferable. However, in the later part of the design phases specific attacks can be used to generate attack tree.

We propose a Security Risk assessment via attack trees, for evaluation of logical combination of attacks, based on the following principles:

a) For considering OR combination of attacks, we will evaluate the canonical form (DNF-Disjunctive Normal Form) of the attack tree. In canonical form we get all the attack paths related with OR (disjunction) relation, to an attack state.
b) Each attack path consists of several basic attack steps related with AND (conjunction). This means the attack path will be successful if and only if all the basic attack steps involved are successful.
c) For considering AND combination among basic attack steps, we evaluate the final likelihood as same as the lowest of the threat level, as this is the critical threat and system cannot be compromised until this threat succeeds.
d) For considering OR combination, the canonical form gives all attack paths to each state related with OR (disjunction) and we do the risk assessment for each attack paths individually.
e) For considering Sequential AND attacks, same as AND, we did not consider impact of sequential dependence on threat level (likelihood). The reachability score for following attacks can be taken as preceding attack, as this provides a gateway to the following attacks.

3.5.2 An example of Security Risk Assessment with Attack Trees

The Attack Tree shown in Figure 23 has been constructed (by our judgement for possible attack scenario) from the threats listed in [40]. Since the attack tree lists specific attacks instead of generalized threats (STRIDE category), the analysis is performed in a later phase of the design process, and basic security strategies have been already adopted to secure system from basic possible attacks.

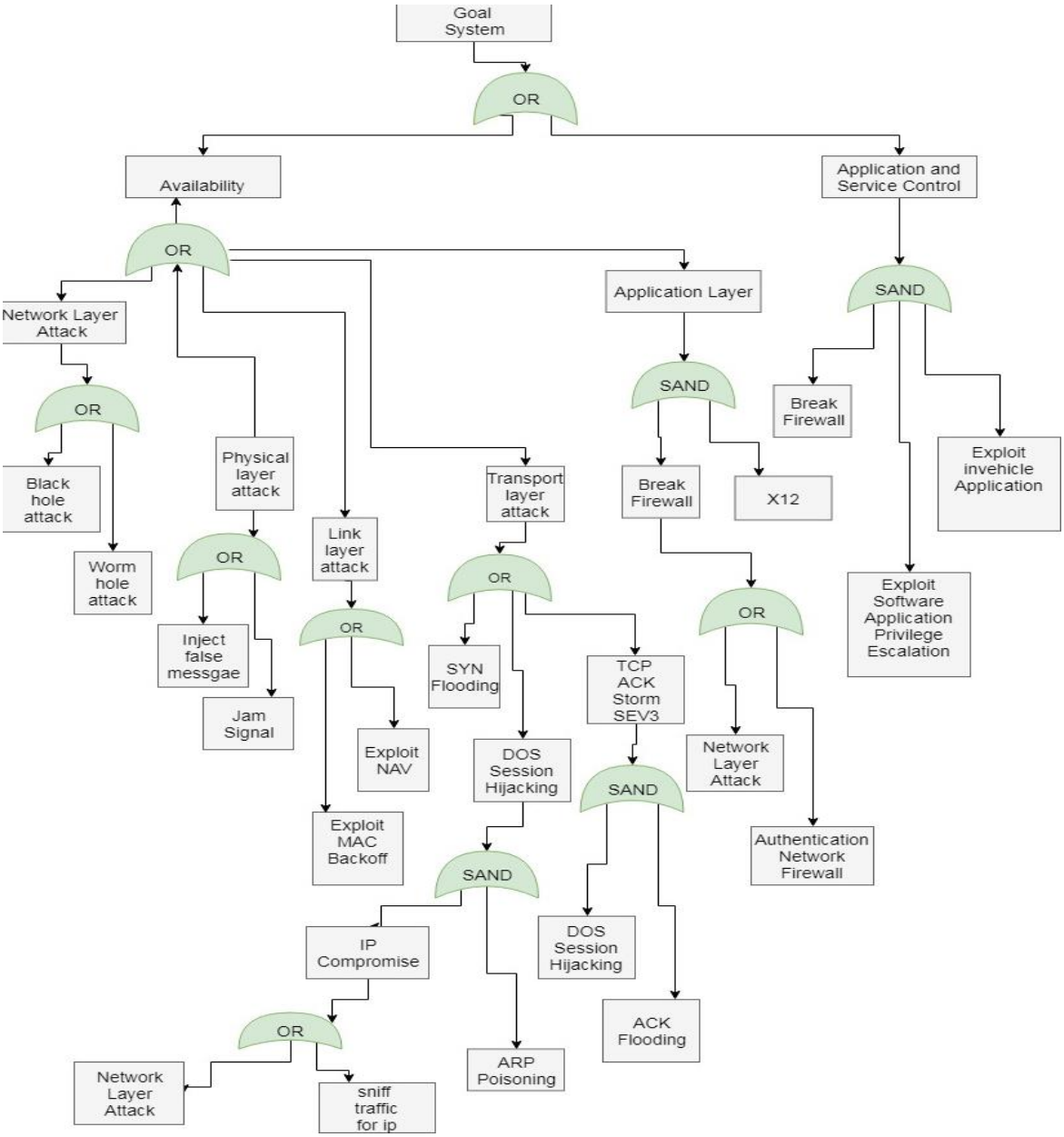


Figure 23 An attack tree for Security Risk Assessment

Steps for risk assessment using the ADtool

Step1 – The attack tree is drawn with help of ADTool, as shown in Figure 24.

Step2 – The model is imported to VBA, Excel, as an .xml file. The VBA scripts sorts basic attack steps (which are at bottom of attack tree) and higher attack states.

Step3 – The evaluator needs to provide scores for threat level parameters and Impact level.

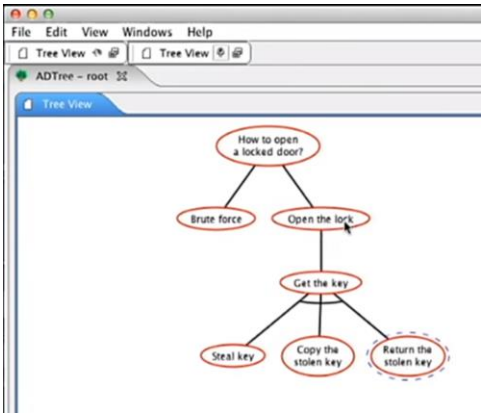


Figure 24 An attack tree obtained using ADT tool

Step4 – The VBA scripts then evaluate, for all higher attacks, which are the basic attack steps involved and how they are related using 'AND', 'OR', 'SAND', as shown in Figure 25.

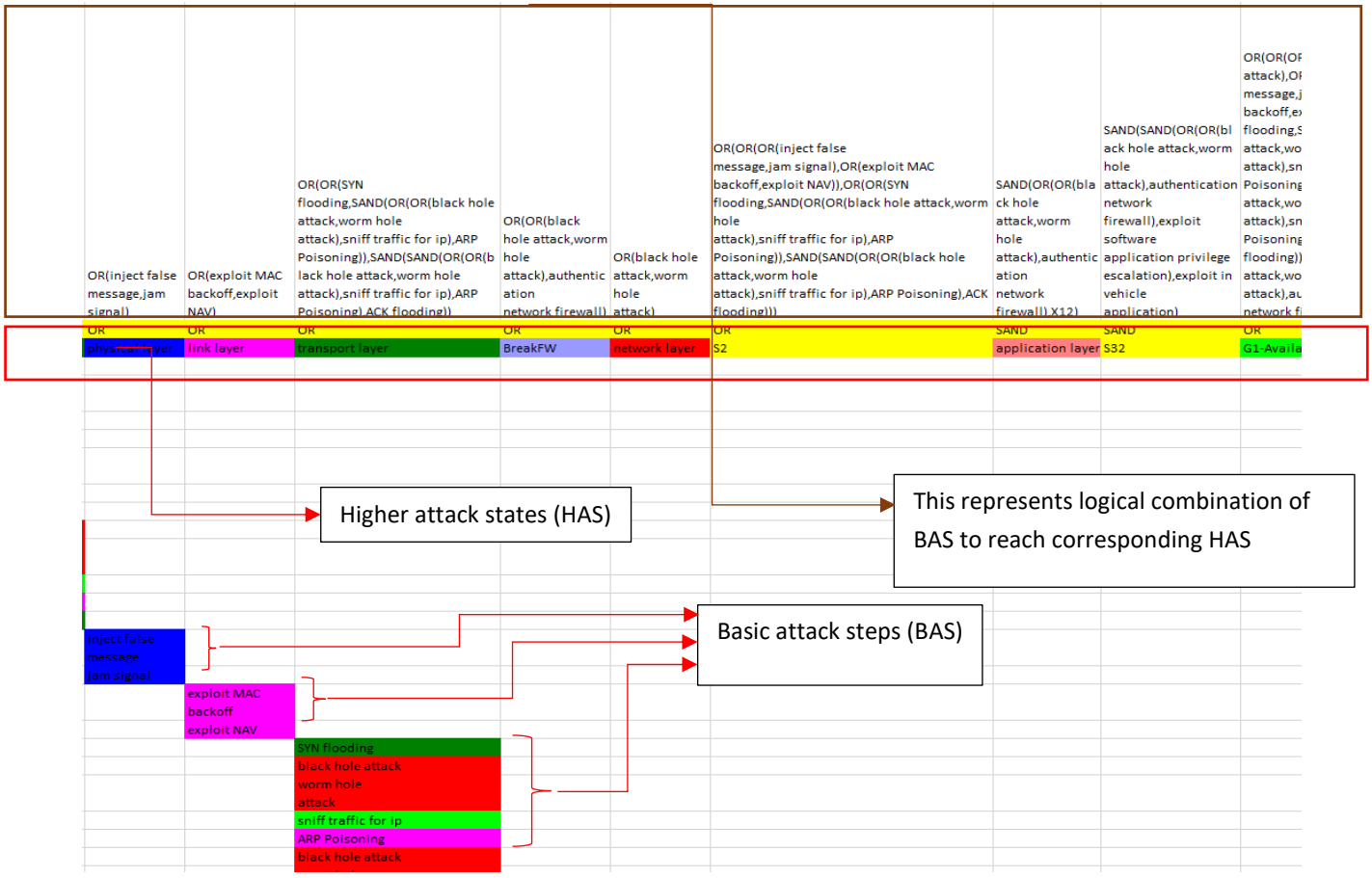


Figure 25 Basic attack steps and relations

Step5 – The implemented scripts evaluate canonical form (disjunctive normal form) for all higher attack states as shown in Figure 26, using the information from Figure 26. For example, the higher attack state 'G1' consist of three attack paths, these attack paths are related to each other with disjunction (HAS will be compromised if any of the three attack paths are successful). The basic attack steps inside each attack path are related with conjunction or sequential conjunction, i.e. attack path will succeed only if all the basic attacks are successful.

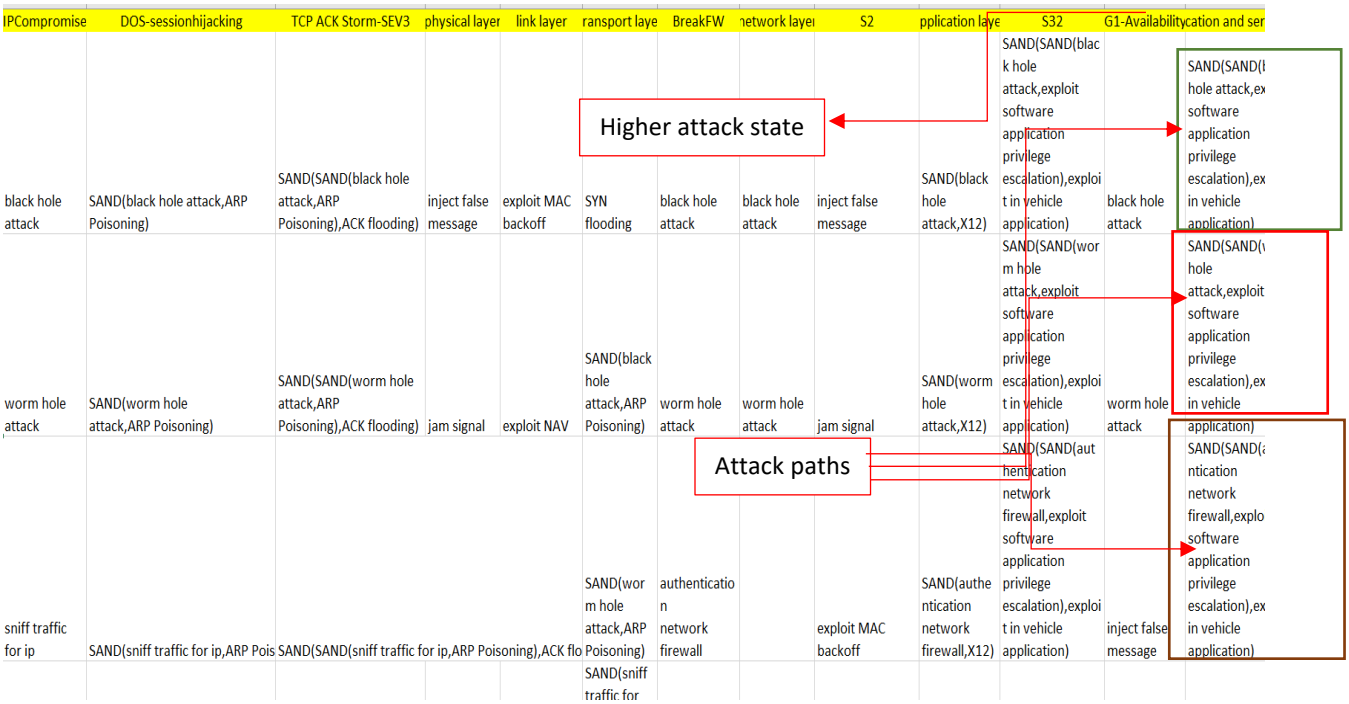


Figure 26 Disjunctive Normal Form for all higher attack states

Step 6 – Considering strategy listed in section 7.1, the critical basic attack and its likelihood will be evaluated for each attack path. Finally, the risk matrix will be obtained using *risk = likelihood x impact*, shown in Figure 27.

	IPCompromise	DOS-sessionhijacking	TCP ACK Storm-SEV3	physical layer	link layer	transport layer	BreakFW	network layer	S2	application layer	S32	G1-Availability	Application and service control	GOAL-SYSTEM
sniff traffic for ip	6	12	18			6			6			6		6
ARP Poisoning		18				9			9			9		9
ACK flooding			21			7			7			7		7
inject false message				16					8			8		8
jam signal				12					6			6		6
exploit MAC backoff					16				8			8		8
exploit NAV					16				8			8		8
SYN flooding						9			9			9		9
authentication network firewall							14							
black hole attack	11						22	22				11		11
worm hole attack	12						24	24				12		12
X12										21		7		7
exploit software application privilege escalation														
exploit in vehicle application											24		6	6

Figure 27 Calculated Risk Matrix

The entries of the matrix represent the risk scores, the row entry tells us which basic attack step is critical, and column entry shows for which higher attack state.

3.5.3 Safety Risk Assessment

There is a number of safety risk assessment methodologies, each having certain benefits and certain limitations. For example, the limitation of the Failure Mode and Effect Analysis is that it is more suited to analyzing systems which have no redundancy and no multiple failures. Fault Tree Analysis (FTA) on the other hand allows logical combination of events but is generally qualitative, can be very complex to be solved analytically and doesn't support repair events. Continuous Time Markov analysis supports repair events but can cause state explosion

problem for complex systems and supports only exponential probability distribution functions. Therefore, hybrid complex systems such as standby switching systems, where the main system, switching system and standby system each has different failure distribution, it is very complex to solve analytically. In such cases Stochastic Colored Petri Net (SCPN) models with simulation are used, they also take care of state explosion problem. Therefore, for redundant systems and complex dynamic systems (switching systems) we should use SCPN model and simulation, such as TimeNET tool.

A complex system however can use individual benefits of each of the methodologies. Such as the highly redundant and complex part of the system can be independently analyzed with SCPN, and the equivalent system can be replaced by a single element with corresponding failure rate. The modified system can be analyzed with the help of FTA.

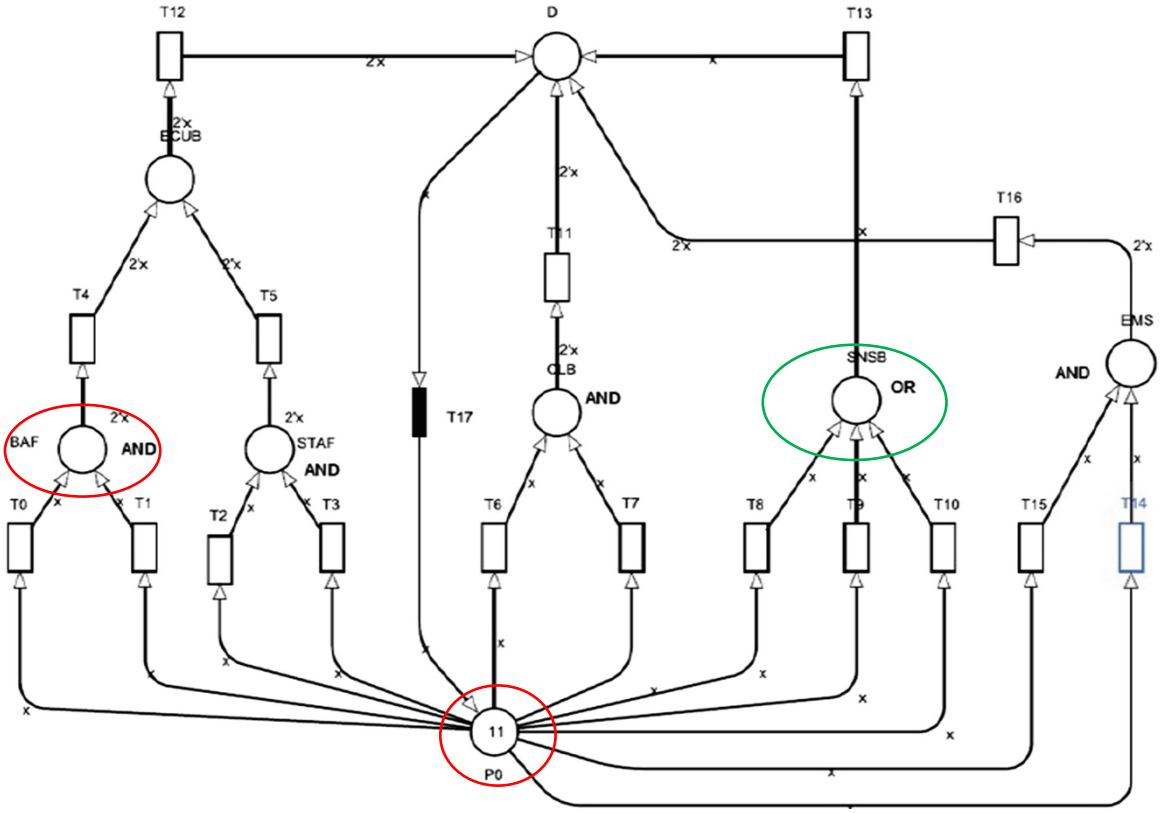


Figure 28 A stochastic coloured Petri net

For example, the above system consists of 11 components, which is represented as ‘11’ tokens present at place P0 in figure 28. Starting from left side, the two components are in conjunction, i.e. the failure will propagate, if and only if both components fail. The rectangular boxes represent the transition rate (failure rate). ‘D’ represents the state when the system is down. Mean time to failure to state ‘D’ can be evaluated using steady state simulation.

4. Platform Layer Tools and Methods

In this section we focus on IoT4CPS WP3 contributions for improving dependability on a CPS platform layer. Specifically, we focus on a method for improving system resilience – Self-Healing by Structural Adaptation.

4.1 Self-Healing by Structural Adaptation

Failed observation data may compromise system's safety. For instance, an erroneous detection of surrounding vehicles that is input to the path planning unit of an autonomous vehicle might cause fatal consequences. Traditional fault-tolerance aims to overcome such critical failures.

Self-healing adapts the system during runtime to mitigate failures. Adaptation is a challenging process that can be summarized by four abstract steps:

- collect environment information and derive internal system properties (state estimation)
- analyze the observations (failure detection)
- decide how to adapt to reach a desired state (find a recovery strategy, e.g. using an ontology)
- act (recover)

Self-healing is the ability of the system to react also to failures not specifically considered during design-time, e.g., faults caused by functional, environmental or technological changes or zero-day malware. A very promising approach of achieving self-healing is through structural adaptation (SHSA), by replacing a failed component with a substitute component by exploiting implicit redundancy (Fig. 29) [24][25].

The implicit redundancy takes a different approach, compared to explicit redundancy which is achieved by duplicating critical system components and voting over the value of the result. Implicit redundancy is the principle of extracting the missing information about a CPS property from related properties. For instance, the property a, provided by service A, can be substituted by *a'*, which is derived from the combination of properties b and c.

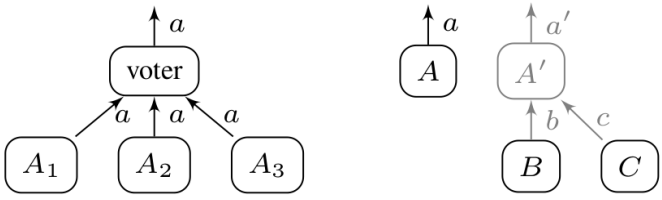


Figure 29: Types of redundancy - explicit (left) and implicit (right) [25].

The algorithm uses a knowledge base, modeled as an ontology which defines the interrelation of properties in the CPS as well as additional runtime information of the CPS.

SHSA can be used to replace failed observation data. It monitors and substitutes CPS variables (cf. signals) in messages communicated between application components (e.g., sensors and controllers) based on a knowledge base. SHSA can be encapsulated in one or more components listening and acting on the communication network of the IoT. The detection identifies a failed component by comparing its output to related information on the network using the relations encoded in the knowledge base. A failed component may be removed or shut-down to avoid faulty messages and possibly propagating the failure. Then SHSA spawns a new component – the *substitute* for the failed one. The substitute subscribes to related information and combines these (again by using the relations in the knowledge base) to provide the needed information.

In this project we specifically target extensions to the knowledge base and substitution, the architectural requirements regarding security and the fault detection. Preliminary results include:

- Guided search of a substitution (speed-up of the recovery process). Evaluation and selection of a substitution extracted from the knowledge base [26].
- Architectural requirements for SHSA and considerations w.r.t. security (will be reported in D6.1).
- Extensions to the knowledge base: encode relations in Prolog (rule-based knowledge base to enable requirements checks). Implement state-aware relations (formerly only state-less relations were possible). Demonstration of how to handle time in relations.
- Fault detection: automatic generation of a runtime monitor for related information considering availability of the information and possible time delays (e.g., latency, physics).

Related work is presented in [27] and D2.1: "*Consolidated state-of-the art report*". Case studies and examples can be found in [25], [27] and D6.1.1: "*Architecture for safe and secure automated driving platform demonstrator"*.

4.2 Architectural Requirements of SHSA

The Architectural Requirements of SHSA are already reported in D3.3. We provide a short reference here, for the sake of completeness.

#	SHSA Requirement	Rationale
1	Dynamic Composability	Add substitutes, remove/replace faulty components.
1a	Reconfigurable Information Flow	Reconfigurable sender/receiver of messages.
1b	Common Communication	One interface to access information.
1c	Freedom of Interference	Adding a substitute shall not alter the system.
1d	Fault Containment	Avoid fault propagation.
2	Information Access	For learning, monitoring and substitution.

Table 6: Requirements to deploy SHSA on a CPS.

The system shall be dynamically composable out of several subsystems or components while preventing side effects or undesired emergent behavior. This requirement enables independent development of system components, reusability of components, and reconfigurability of the system during runtime. In adaptive CPS we need dynamic composability in order to be able to add, change or remove components during runtime.

By further refining the dynamic composability requirement we get the following:

- Reconfigurable Communication: The flow of information shall be reconfigurable. Services share information, hence, when a service is added or removed, the communication to and from the service must be installed.
- Common Communication Interface: Since the system is assembled out of several subsystems, a common communication interface must be provided to the reconfiguration mechanism. Hence, SHSA is a challenge especially in heterogeneous systems.
- Freedom of Interference: The reconfiguration mechanism shall not compromise the system's functionality.
- Fault Containment: A failure of a service shall not propagate.

Information Access requirement relates to enabling access to the information related to system state.

5. Network Layer Tools and Methods

In this section we focus on IoT4CPS WP3 contributions for improving dependability on network CPS layer. Specifically, we focus on a recommender systems which should help the IoT system architects to select the protocols which meet the requirements of their systems.

5.1 Recommender System for Dependable IoT applications

Along with the growing market of Industrial IoT (IIoT) applications, the set of available network technologies is continuously expanding. Today, developers have a huge set of connectivity networks at their disposal, ranging from short-range networks such as Bluetooth to global connectivity via satellite networks [1]. Figure 30 tries to give an overview of existing technologies while not claiming to be exhaustive. Depending on the specific use case of each IIoT application, different approaches constitute the most cost-effective network technology solution, as there is no “one-size-fits-all” solution.

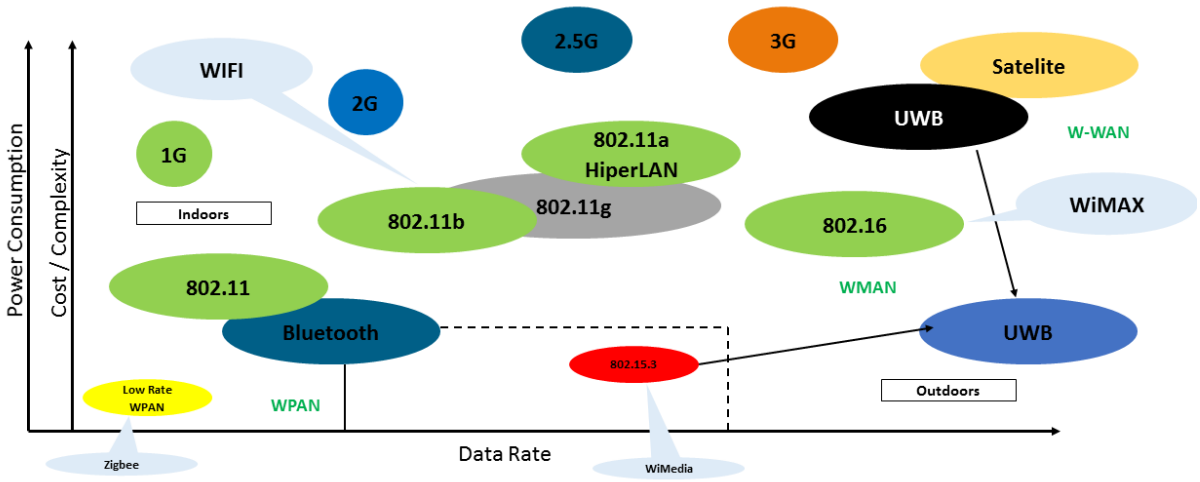


Figure 30:Technology overview [1], [2], [3], [4]

Choosing the set of network technologies, which fits the needs of the IIoT use case must be a careful trade-off between the ability of the technologies to meet specific functional requirements and the related costs [1]. Complex IIoT systems — as sketched in Figure 31 – have a significantly larger set of dependability requirements compared to “normal” IoT applications. As the systems connectivity network plays a major role in fulfilling these requirements, choosing the correct set of technologies is crucial.

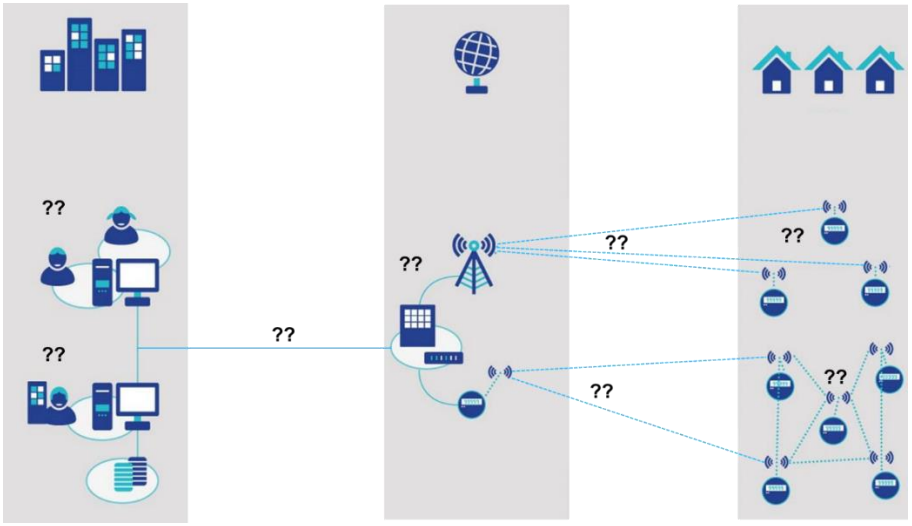


Figure 31: A complex IIoT system comprising file, edge and cloud components

5.1.1 Aim of the Recommender System

Depending on the application and its scope, different system architectures can be applied. However, most applications follow the generic system architecture. This architecture comprises different system levels and possible interconnections.

The challenge in building such systems lies on interconnecting already ongoing engineering activities and brownfield devices at multiple levels and enrich them with tools to address dependability aspects of the system. Usually such a system design is derived by experts and causes a lot of effort and state-of-the-art expertise. The proposed approach tries to perform the design and partial configuration of a system in a semi-automatic way. Thereby, specific system constraints and requirements will be considered. They are ranging from basic communication properties such as energy consumption, bandwidth and latency to specific dependability attributes such as integrity and availability.

The system designer has to provide application specific requirements (e.g., purpose, location, connectivity, power supply) and high-level architecture patterns (e.g., direct connection field cloud or multilevel communication via edge devices). Based on this information, the recommender system is capable of computing a feasible system setup (system topology, technologies to apply within the system) regarding a specified use case, without the need of consulting experts. Figure 32 illustrates the workflow of the recommender system.

Beside generating a suggested system topology, the recommender system will match the application's demand regarding dependability attributes with the offered attributes of the possible technologies. Thereby, qualitative and quantitative measures are applied. Based on this, a final selection of technologies is possible.

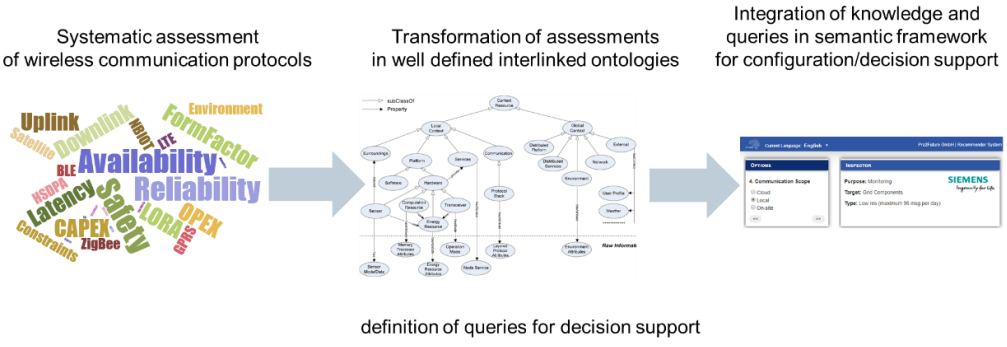


Figure 32: Illustration of the workflow of the recommender system

5.1.2 Knowledge Base

The aimed ability of the recommender system is based on maintaining a database, comprising of the technical and functional characteristics of all the available network technologies. The database of the recommender system is based on the Open Semantic Framework (OSF). The open semantic framework, cf. [34], structures information in domain-specific knowledge packs (KP), which depend on several core ontologies (containing general information about concepts, quantities, units, events etc.). The OSF allows for a modular management of knowledge specifically tailored to the application as sketch in Figure 33. OSF knowledge can be accessed and managed during runtime via a REST API and thus easily integrated in established process environments. Using construct or update features, it is possible to curate and extend both the available knowledge in the OSF as well as construct or update the corresponding queries. Additional domain-specific knowledge packs and domain specific queries can be added to extend the functionality of the framework towards the desired application domain.

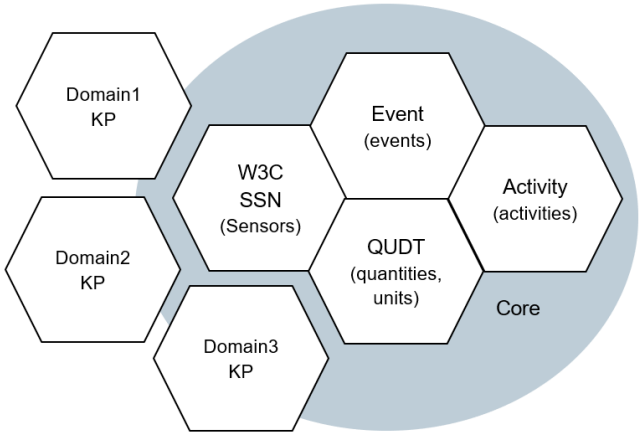


Figure 33: Modular architecture of the Open Semantic Framework

Based on the OSF different KP were developed as basis for the recommender system as depicted in Figure 34. The “protocol KP” comprises the basic properties of communication technologies. Besides that, the KP focusses on measures regarding dependability. These measures contain a set of dependability methods such as multipath routing, frequency hopping, or retransmit mechanisms. In addition to the method, also the layer (w.r.t. the ISO-OSI reference model) is specified. The dependability measure of the protocol KP is further linked to concerned threats of the dependability KP. Threats are basically described in terms of affecting dependability attributes and their impact on the overall system. The impact in this KP is agnostic w.r.t. to application domains or systems.

If a more complex modelling is necessary, a KP based on the threat model introduced in Section 4 and 5 can be designed. With this model a higher level of detail can be achieved than in the basic model. The OSF enables to curate dependability threats independently from protocols (different knowledge packs). Additionally, the links across different knowledge packs are used to assess how well specific protocol instances can deal with threats. This allows an assessment of the overall dependability of a set of protocols across different threat levels on different layers.

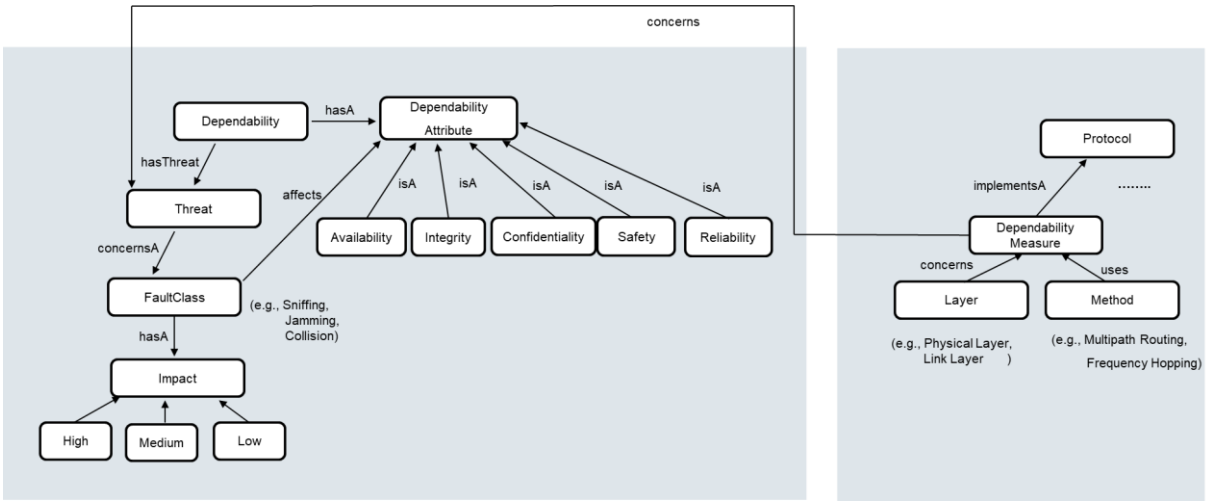


Figure 34: Core KP for the recommender system

5.1.3 User Interface

A first sketch of a user interface is depicted in Figure 35. On the left-hand side, it allows the system designer to select the environment a high-level requirements. According to this selection, the centre of the UI reflects the architecture of the overall system. Additionally, it is possible to set specific numerical parameters for the overall system on the right-hand side. The bar on the top finally depicts considered. According to the traffic-light colouring, the technologies are justified.

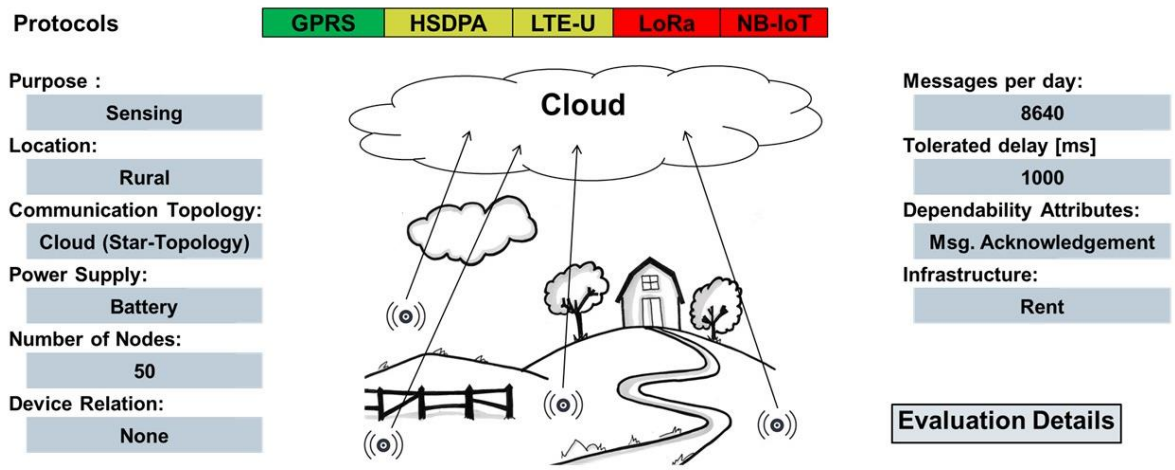


Figure 35: Illustration of the Recommender System with possible input parameters

5.1.4 Examples

We present two examples that illustrate the selection of protocols according the specific requirements. While the first example aims at designing a conventional industrial IOT system, the second example illustrates a configuration with dependability demands. In both examples, prerequisites regarding the availability of KP are stated. Depending on the aim of the design and the level of details, different KPs are necessary.

5.1.4.1 Identification of suitable communication protocols

- Prerequisites
    - Knowledge-pack containing specifications of wireless protocols:

        - Network-Type
        - Bandwidth
        - Latency
        - Energy Consumption
        - Coverage
        - Number of supported Nodes
        - Infrastructure Constraints
        - Dependability Assessment
    - Knowledge-pack containing options for recommendation:
        - Purpose (monitoring, control)
        - Communication scope (local, local-aggregation2cloud, cloud)
        - Use-case environment
        - Amount of devices
        - Amount of required transmissions
        - Expected size of transmissions
        - Tolerable delay (linked with communication use-case)
        - Own Infrastructure
        - Required life-span of devices
        - Operation Cost
- Example Input
    - **Purpose:** Monitoring
    - **Communication scope:** Local-Aggregation2Cloud
    - **Use-case environment:** Urban
    - **Amount of devices:** 200 sensors per aggregator
    - **Amount of required transmissions:** 480 (sensor / aggregator), 96 (aggregator / cloud)
    - **Expected size of transmissions:** < 10 Byte
    - **Tolerable delay:** < 200 ms
    - **Own Infrastructure:** unspecified
    - **Required life-span of devices:** > 1 year
    - **Operation Cost**: <1€ / year
- Example Output
    - Potential Solutions Sensor-Aggregator:
        - **BLE 802.15.1, 802.15.4**
        - **802.15.4e, 802.11**
    - Potential Solutions Aggregator-Cloud:
        - **NB-IoT, LoRaWAN**

5.1.4.2 Identification of suitable communication protocols including dependability aspects

- Prerequisites:
    - Knowledge-pack containing specifications of wireless protocols:
    - Adapted Options Knowledge-pack including dependability of solution:
        - Dependability Requirement
    - Knowledge-pack containing available protocol dependability features
        - Confidentiality (protection against, sniffing, tampering, eavesdropping)
        - Integrity (protection against sinkhole, spoofing, relay attacks)
        - Availability features (protection against jamming, desynchronization, packet/connection loss)
- Example Input

- **Purpose:** Monitoring
- **Communication scope:** Local-Aggregation2Cloud
- **Use-case environment:** Urban
- **Number of devices:** 200 sensors per aggregator
- **Amount of required transmissions:** 480 (sensor / aggregator), 96 (aggregator / cloud)
- **Expected size of transmissions:** < 10 Byte
- **Tolerable delay:** < 200 ms
- **Own Infrastructure:** unspecified
- **Dependability Requirement: High**
- **Required life-span of devices:** > 1 year
- **but Operation Cost: >1€ / year**

- Example Output
  - Potential Solutions Sensor-Aggregator: **BLE 802.15.1 Mesh Topology**
    BLE supports channel hopping, which mitigates the impact of other 2.4Ghz technologies on the communication. A mesh topology further strengthens the dependability of the solution
  - Potential Solutions Sensor-Aggregator: **LTE**
    Given the high dependability requirements the solutions previously recommended (LoRaWAN, NB-IoT) are no longer suitable. While LTE leads to higher costs it allows for a higher dependability

6. Conclusion

Deliverable "*D3.2: Guidelines, processes and recommendations for the design of dependable IoT Systems"* provides a detailed insight into state-of-the art tools and methods, which are the outcome of IoT4CPS project WP3. The methods and tools are strongly motivated by our two main use cases of Automated Driving and Smart Production. The aim of methods and tools of D3.2 is to build dependable systems by improving their safety & security using:

- Security tools such as ThreatGet, GSFlow, Moreto
- Security risk assessment methods such as FMVEA
- Co-engineering methods for safety and security
- Resilience techniques and architectures, such as Self-Healing by Structural adaptation
- Trusted localization and orientation
- Recommender Systems for building large IoT-based applications
- Sensor-level security
- High-level guidelines on writing crypto APIs
- V&V patterns for Security Risk Assessment

Our tools and methods cover a wide spectrum of both design time and runtime methods which aim to make the system more safe, secure, resilient to failures, trustworthy and reliable. We strongly believe that dependability is a complex system property which must be addressed at different levels of complexity and multiple stages of product lifecycle. IoT4CPS consortium is looking forward to further improving our tools and methods in order to provide the industrial partners the means to achieve dependable IoT systems.

7. Appendix: V&V pattern – Security Risk Assessment with Attack Trees

In this appendix we provide a formalized abstraction, namely a V&V pattern, for our Security Risk Assessment from section 7.1.1.

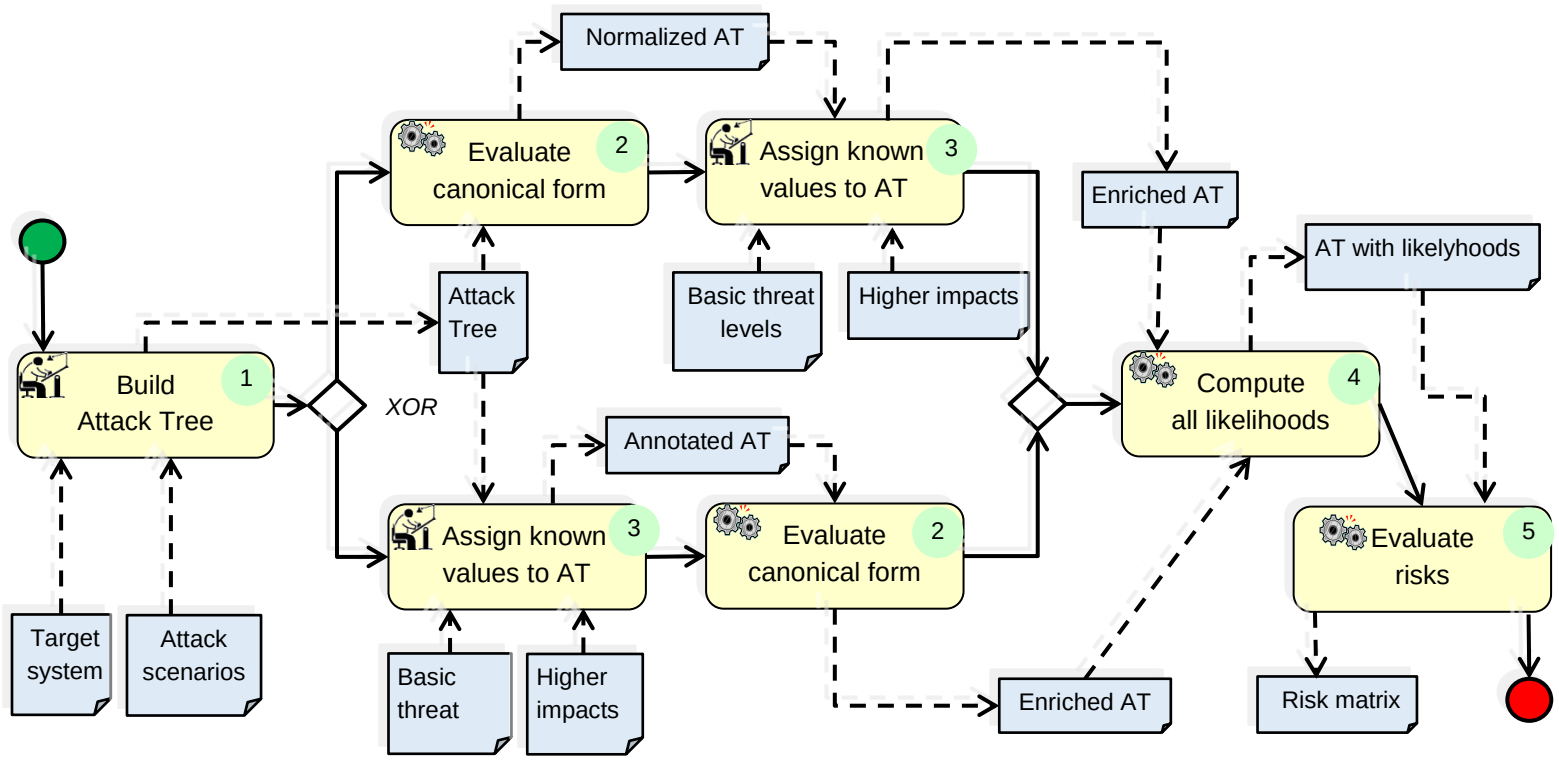


Figure 36: A pattern for Security Risk Assessment

The definitions from the pattern include:

- Attack Tree (AT): a tree graph with inner nodes representing logical OR/AND/SAND operations of lower level tree elements
- Basic Attack (BA): bottommost AT nodes, representing atomic or elementary security threats
- Higher Attack State (HAS): all not BA-nodes in the AT.
- Impact (Strength) / Severity Level: severity of consequences of a successful attack
- Likelihood / Threat Level: probability of occurrence of a certain threat or attack
- Risk: Impact * Likelihood
- Target System (TS): the CPS for which security risks shall be analyzed

Participants and important artefacts include:

- System Expert: a person that has sufficient knowledge about the TS for providing all relevant information to the Security Expert
- Security Expert: a person who can identify all BAs relevant for the TS, and who knows how they interrelate for building the AT.
- Target System Description: information at sufficient detail about the TS that can be used by the experts for building the appropriate AT.

- Attack scenarios: considering known attack scenarios help avoiding omission of security threats.
- Attack Tree: tree-like graph representing derivation of complex threats from basic attacks.
- Basic Threat Levels: likelihoods of BAs.
- Higher Impacts: impacts of higher nodes in the BA tree (that are known a priori).
- Normalized AT: AT turned into disjunctive normal form (DNF).
- Enriched AT: normalized AT, annotated with basic threat levels and higher impacts.
- AT with Likelihoods: enriched AT with likelihoods computed for all nodes.
- Risk Matrix: matrix showing the risks of all attacks for all target system. components/items.

Actions and collaborations include:

(1) Build Attack Tree:
   - Identify the BA that are principally possible for the given TS.
   - Multiple attack scenarios are considered by combining BAs via 'OR', 'AND', and 'SAND' (sequential AND) nodes, reflecting experts knowledge about their combinatorial characteristics.
   - Multi-stage attacks are considered by iterating step 2, including new nodes, in a tree-like manner until a topmost attack threat is reached.

(2) Evaluate canonical form: For considering OR combination of attacks, evaluate the canonical form (DNF – Disjunctive Normal Form) of the attack tree. In canonical form all the attack paths are related to an attack state with OR (disjunction) relations. Now, each attack path consists of several basic attack steps related with (S)AND ((sequential) conjunction). This means the attack path will be successful if and only if all the basic attack steps involved are successful.

(3) Assign known values to AT: assign ordinal (e.g. "very low" 0 … "very high" 3) threat levels to BAs', and quantitative severity levels to all higher nodes of the AT.

(4) Compute all likelihoods: For the higher level and multi-level attacks represented (HAS) by the non-leave nodes of the AT, their likelihoods are computed as follows:
   - For OR nodes, the canonical form gives all attack paths to each state related with OR (disjunction) and we do the risk assessment for each attack paths individually. Note that the resulting risks are propagated separately to the next higher level.
   - For AND nodes, likelihood is the minimum of all direct lover level nodes.
   - For SAND nodes, compute likelihood as for AND nodes (i.e. do not consider the impact of sequential dependences on threat level). (The reachability score for following attacks can be taken as preceding attack, as this provides a gateway to the following attacks.)
   - Note: a OR-HAS will be compromised if any of the its attack paths is successful.

(5) Evaluate risks (risk matrix): For each node, compute risk as product likelihood*impact. Represent the result in a matrix with all TS components on one axis, and all threats – basic as well as higher and multi – on the other.

8. References

[1] Frederic Vannieuwenborg, Sofie Verbrugge and Didier Colle. “Choosing IoT-connectivity? A guiding methodology based on functional characteristics and economic considerations”. Transactions on Emerging Telecommunications Technologies(2018;29:e3308)

[2] Elkhodr, Mahmoud, Seyed A. Shahrestani, and Hon Nin Cheung. "Emerging wireless technologies in the internet of things: a comparative study." International Journal of Wireless and Mobile Networks 8.5 (2016): 67-82.

[3] Tardy, Isabelle, et al. "Comparison of wireless techniques applied to environmental sensor monitoring." SINTEF Report (2017).

[4] Raza, Usman, Parag Kulkarni, and Mahesh Sooriyabandara. "Low power wide area networks: An overview." IEEE Communications Surveys & Tutorials 19.2 (2017): 855-873.

[5] "MulteFire™ lights up the path for universal wireless service." https://www.multefire.org/wp-content/uploads/2016/10/72-multefire-lights-up-the-path-for-universal-wireless-service.pdf. Accessed 4 Dec. 2018.

[6] Comparing IoT Technologies at a Glance | Wi-SUN Alliance." https://www.wi-sun.org/news/comp-iot-tech/. Accessed 4 Dec. 2018.

[7] Long Range Wireless IoT | 2018 Guide to LoRa and Other LPWAN ...." https://www.postscapes.com/long-range-wireless-iot-protocol-lora/. Accessed 4 Dec. 2018.

[8] "Narrowband IoT - Wikipedia." https://en.wikipedia.org/wiki/Narrowband_IoT. 4.12.2018.

[9] "Cellular IoT – Part 9 – 50.000 devices per cell – WirelessMoves." 18 Sep. 2016, https://blog.wirelessmoves.com/2016/09/cellular-iot-part-9-50-000-devices.html. 4 Dec. 2018.

[10] Naik, Nitin. "LPWAN technologies for IoT systems: choice between ultra narrow band and spread spectrum." 2018 IEEE International Systems Engineering Symposium (ISSE). IEEE, 2018.

[11] Mikhaylov, Konstantin, Juha Petaejaejaervi, and Tuomo Haenninen. "Analysis of capacity and scalability of the LoRa low power wide area network technology." European Wireless 2016; 22th European Wireless Conference; Proceedings of. VDE, 2016.

[12] Chen, Min, et al. "Narrow band internet of things." IEEE Access5 (2017): 20557-20577.

[13] Talwar, Shilpa, et al. "Enabling technologies and architectures for 5G wireless." Microwave

[14] Muhammad A. Iqbal, Oladiran G.Olaleye and Magdy A. Bayoumi,”A Review on Internet of Things (Iot): Security and Privacy Requirements and the Solution Approaches”. Global Journal of Computer Science and Technology: ENetwork, Web & Security - 2016.

[15] Kejun Chen, Shuai Zhang, Zhikun Li, Yi Zhang, Qingxu Deng, Sandip Ray and Yier Jin,”Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice”. Journal of Hardware and Systems Security - 10 May 2018.

[16] S. M. Riazul Islam, Daehan Kwak, Md. Humaun Kabir, Mahmud Hossain And Kyung-Sup Kwak,”The Internet of Things for Health Care: A Comprehensive Survey”. IEEE Access - June 1, 2015.

[17] Rashmi Sharan Sinha, Yiqiao Wei and Seung-Hoon Hwang,”A survey on LPWA technology: LoRa and NB-IoT”. Korean Institute of Communication and Information Sciences. March 14, 2017.

[18] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid.” Internet of Things in the 5G Era: Enablers, Architecture, and Business Models”. IEEE Journal on Selected Areas in Communications, Vol. 34, No. 3, March 2016.

[19] The Future of Connectivity in IoT Deployments, Deloite Report.

[20] Richard Harada & Edgar Sotter,”Automated Monitoring for Inspections and Condition-based Maintenance- Part III: Industrial IoT Networks”.

[21] Dong-gil Kim, Seawook Park, Kyungmin Kang and Dongik Lee,” A Deterministic Wireless Network For Feedback Control Based On Ieee 802.15.4”, School of Electrical Engineering and Computer Science, Kyungpook National University 1370 Sankyug-dong, Buk-gu, Daegu, 702-701, Korea - April 21, 2016.

[22] Kay Romer and Friedemann Mattern, “The Design Space of Wireless Sensor Networks”. IEEE Wireless Communications • December 2004.

[23] C.Schmittner, Z. Ma, E. Schoitsch, T. Gruber, „A case study of FMVEA and CHASSIS as safety and security co-analysis method for automotive cyber-physical systems”

[24] Oliver Höftberger. Knowledge-Based Dynamic Reconfiguration for Embedded Real-Time Systems. PhD thesis, TU Wien, Institute of Computer Engineering, Wien, 2015.

[25] D. Ratasich, O. Höftberger, H. Isakovic, M. Shafique, and R. Grosu. A Self-Healing Framework for Building Resilient Cyber-Physical Systems. In 2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC), pages 133-140, May 2017.

[26] D. Ratasich, T. Preindl, K. Selyunin, and R. Grosu. Self-healing by property-guided structural adaptation. In 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pages 199-205, May 2018.

[27] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci. A Roadmap Towards Resilient Internet of Things for Cyber-Physical Systems. IEEE Access, pages 1-24, Jan 2019.

[28] A.Shostack, Threat modeling: Designing for security. John Wiley & Sons, 2014.

[29] IEC. Industrial communication networks - Security for industrial automation and control systems, IEC 62443-4-2. url: https://webstore.iec.ch/publication/34421.

[30] https://satoss.uni.lu/members/piotr/adtool/

[31] https://github.com/IEEM-HsKA/AAD

[32] https://siddhathrokin.wixsite.com/website

[33] Thomas L. Satty: The Analytical Network Process, 2006

[34] [34] CENELEC, European Committee for Electrotechnical Standardization: EN 50126 Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1: Basic requirements and generic process (1999)

[35] CENELEC, European Committee for Electrotechnical Standardization: EN 50128 Railway applications – Communication, signalling and processing systems – SW for railway control and protection Systems (2011)

[36] CENELEC, European Committee for Electrotechnical Standardization: EN 50129 Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling (2003)

[37] SAE International: J3061 Cybersecurity Guidebook for Cyber-Physical Vehicle Systems (1 2016)

[38] International Electrotechnical Commission: IEC 62443: Industrial communication networks Network and system security

[39] Avizienis, Algirdas, et al. "Basic concepts and taxonomy of dependable and secure computing." IEEE transactions on dependable and secure computing 1.1 (2004): 11-33.

[40] Pooja Mishra; Anil Jaiswal: A Review on Safety Mechanisms in Vehicular Ad hoc Network, 2015