Safety-Security Convergence: Automation of IEC 62443-3-2

Mike DA SILVA[a], Stéphane MOCANU[b], Maxime PUYS[a,c], Pierre-Henri THEVENON[a]

[a] *CEA-Leti, Univ. Grenoble Alpes, 17 avenue des Martyrs, Grenoble, 38054, France*
[b] *Laboratoire d'Informatique de Grenoble, Univ. Grenoble Alpes, CNRS, INRIA, Grenoble-INP, Grenoble, France*
[c] *Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines Saint-Etienne, LIMOS, 63000, Clermont-Ferrand, France*

Abstract

Industrial Control Systems (ICS) are designed to provide a service, such as power generation or water treatment, while protecting people, assets, and the environment against hazard. However, ICS now integrate Information Technology (IT) and are interconnected with the outside world such as the Internet, thereby exposing their infrastructures to cyberattacks. Cyberattacks have thus become new threats for industrial system operations and, more specifically, for their safety. To address the issue, this paper presents a comprehensive cybersecurity risk assessment for the safety of ICS. We apply our method to automate industrial cybersecurity risk assessment as specified in the recent (2020) IEC 62443-3-2 standard, which is widely used in the industrial cybersecurity domain. By automating parts of these risk assessment processes, we can reduce the error-prone manual efforts and increase the consistency of risk assessment. More specifically, the proposed risk assessment comprises three parts which, respectively: (1) identify the specific vulnerabilities of industrial control systems, (2) determine the attack scenarios that compromise the safety of the system and (3) assess whether the attack scenarios are tolerable by the organization's policy. In the first part, we automated the entire threat modeling process of *Micorsoft Threat Modeling Tool* by developing an automatable method for building the system model, in the form of a data flow diagram, from a standard XML file called PLCOpen. This automation of the Micorsoft Threat Modeling Tool process enables us to automate vulnerability identification for industrial control systems. In the second part, we enhance a previous work that generates theoretical safety-compromising attack scenarios by building a complete attack scenario from system vulnerabilities to safety compromise. Finally, in the third part, we rank the attack scenarios using a specific risk matrix in order to determine which scenarios exceed the risk tolerable by the organization and therefore require additional controls.

Keywords: Safety, Cybersecurity, Risk assessment, ICS, IT, OT

1. Introduction

Over the last twenty years, ICS have become increasingly open to the internet through the interconnection of Information and Operational Technologies (IT and OT). However, risk management of OT systems is fundamentally different from that of IT systems, as highlighted in the NIST SP800-82r3 [1] (2023) guide through a comparison of OT and

IT systems. The guide explains that IT and OT have different purposes (Managing data vs. Controlling the physical world), which influences the properties that need to be secured. Indeed, in IT, data confidentiality and integrity are paramount, with the delay of business operations a major risk, whereas Human safety is paramount in OT and availability is critical for the system. These differences highlight the need to use dedicated cybersecurity risk assessment methods for ICS in order to design a system that is safe and secure. We define safety and security according to the IEC 62443-1-1 [2] standard: *Safety* is a property that characterizes the system's ability to protect itself from predictable events while *Security* is the ability to protect the system from cyber threats. In this paper, security and cybersecurity will be used interchangeably.

Cybersecurity risk assessments for ICS are generally based on generic processes defined by standards such as IEC 62443-3-2 [3]. These processes are frequently applied on an ad hoc basis in a succession of manual tasks, prone to human error, the quality of which depends essentially on the team's expertise. In order to reduce the manual effort required to apply the processes defined in cybersecurity standards, we have developed a method, presented in Section 3, that automates an ISO 31000 [4] risk assessment applied to ICS cybersecurity. ISO 31000 [4] provides a generic risk assessment applicable to any organization, regardless of its sector, industry or the form of risk it faces. We therefore propose a method that is sufficiently generic to be integrated into, and partially automate, risk assessment processes such as IEC 62443-3-2.

The method we have designed meets two challenges. The first challenge is to propose a safety-security risk assessment method that scales up to large systems. Indeed, the majority of safety-security risk assessments are designed to provide a detailed attacker and system behavior in order to build complex attack scenarios but lack of applicability for large systems (over 20 sensors and actuators[1]). The second challenge was to design respectively a comprehensive risk assessment. In fact, most of the methods reviewed do not cover the whole process of a risk assessment as defined in the ISO 31000 standard, i.e. identify, analyze and evaluate the risk. The method that we developed integrate each step of identification, analyze and evaluation of a comprehensive risk assessment process. These two criteria, *Scalability* and *Comprehensiveness*, are discussed in section 5 to compare our work with the state-of-the-art.

Contributions. The IEC 62443-3-2 [3] standard defines the requirements needed to assess the cybersecurity risk for ICS. However, this standard is generic and its application is not straightforward. In this paper, we attempt to partially automate the initial and the detailed cybersecurity risk assessments of the IEC 62443-3-2 [3] standard from two methodologies that we previously developed: (i) an extension of a threat modeling tool to identify ICS vulnerabilities [5] and (ii) a methodology that identifies cybersecurity risk for the system's safety [6]. We thus extend our previous work [5, 6], aiming for a comprehensive methodology and improved automation. We make the following contributions:

1. From [5], we automatically build the system model needed for the threat modeling tool from a standard XML file (commonly known as PLCOpen [7]).

[1]This value was determined empirically according to a review of the system modeled in the literature (cf. Table 6)

2. We identify the impact of system vulnerabilities for safety by mapping vulnerabilities generated by the threat modeling tool [5] using the methodology that identifies cybersecurity risk for safety [6].
3. Finally, we put forward a specific risk matrix that assesses cybersecurity risk according to the likelihood of an attack scenario and its impact on the system's safety.

Outline. We first present the background knowledge in Section 2, followed by an introduction to our methodology in Section 3, and an evaluation in Section 4. In Section 5, we compare our work with the state-of-the-art in safety-security risk assessment, as well as the state-of-the-art in automation of the IEC 62443-3-2 risk assessment. Finally, we conclude in Section 6 and identify perspectives for further work.

2. Background

This section briefly describes the elements and concepts used in our methodology. First, we introduce the concepts of *unified* and *integrated* safety and security risk assessment, and explain why we developed an *integrated* methodology. The methodology aims to automate the IEC 62443-3-2 risk assessment, whose background and content we briefly explain. We also present the IEC 61131 and use an XML-based exchange file of PLC programs, written in the SFC and FBD programming languages, to model the system. Both the XML file and the programming languages are specified in this standard. We then describe the ISO 31000 standard we used to build our risk assessment process. Finally, we present the STRIDE methodology that we used to threat model the system.

2.1. Safety-Security Risks Assessments

We find two main trends in the literature with respect to the issue of safety-security risk assessment. The first proposes *unified* [8] assessment of safety and security risks based on a combination of modeling methods incorporated into a single methodology. Conversely, *integrated* [8] methods separate the initial safety and security risk assessments and then define the causal relationships between safety and security.

In this paper, we use an *integrated* method since such methods are able to place safety as the priority property to be ensured and thus security as a property at the service of safety. In our view, coupling safety and security models helps to highlight the consequences and cascading effects on the system, but it also adds to the modeling complexity (e.g., increases the combinatorial possibilities, time to process outputs, etc.) and limits its applicability to an entire system. Moreover, while safety risk assessments are usually probabilistic due to the predictability of failures, this is not the case in cybersecurity which increases complexity in order to unify the safety and security models.

2.2. IEC 62443

IEC 62443 is an international series of standards on "Industrial communication networks - Network and system security". The standard is divided into four parts: (1) General, (2) Policies and Procedures, (3) System, and (4) Component. The component standards IEC 62443-4-1 [9] (2018) and IEC 62443-4-2 [10] (2019) are vendor centric (product supplier and service provider) in contrast to the system standards IEC 62443-3-X [3, 11] which are intended for asset owners, system integrators, product suppliers, service providers, and compliance authorities.

In particular, the IEC 62443-3-2 (2020) standard proposes a security risk assessment for system designs composed of 7 steps referred to as zone and conduit requirements (ZCR) in the standard.The standard's workflow is shown in Figure 1. First, the standard requires identifying the system under consideration (step 1) through a system architecture and an assets inventory. Following this, an initial cybersecurity risk assessment is performed (step 2) to identify cybersecurity risks that could impact critical system operations. The system is then partitioned into zones and conduits (step 3), i.e., into groups of physical assets and communication channels for which a detailed cybersecurity risk assessment is performed (step 5) if the organization states that the initial risk, determined in step 2, exceeds the tolerable risk (step 4); otherwise, a cybersecurity requirement specification is defined (step 6) and approved (step 7). As we will see later in Section 3, our methodology can be used to partially automate the risk assessments of this standard, i.e., the initial (step 2) and the detailed (step 5) risk assessment.

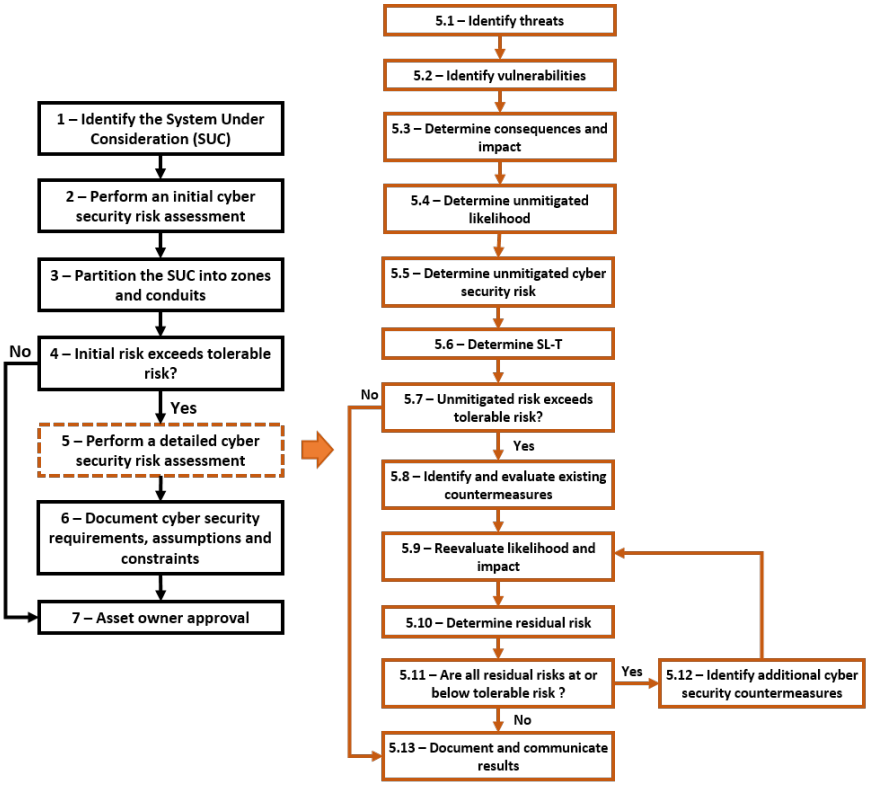

Figure 1: IEC 62443-3-2 Standard Workflow

2.3. IEC 61131

The IEC 61131 [12, 13, 7] standard describes Programmable Controllers (PC), referred to in the standard as Programmable Logic Controllers (PLC) to avoid confusion with Personal Computers (PC), in 10 different parts.

IEC 61131 Part 3 [12] (2013) specifies the syntax and semantics of a suite of PLC programming languages, including two textual languages, Instruction List (IL)[2] and Structured Text (ST), as well as three graphical languages, Ladder Diagram (LD), Function Block Diagram (FBD), and Sequential Function Chart (SFC).

IEC 61131 Part 5 [13] (2000) specifies how PLCs can communicate with other devices, either as a client or as a server. In addition, this part of the standard defines the communication functions of PLCs and the corresponding function blocks (FBs) so that the PLC communication functions can be described using a 61131-3 function block diagram program (FBD).

Finally, IEC 61131-10 [7] (2019) specifies a standard format for exchanging XML-based 61131-3 programs, also known as the PLCOpen format, between different development environments. As shown in Section 3, we will use this format to extract both the control code of the PLC written in IEC 61131-3 programming language and the PLC communication functions specified in IEC 61131-5.

2.4. ISO 31000

ISO 31000 [4] is an international standard that provides risk management guidelines and offers a generic approach applicable to any organization, regardless of its sector, industry, or the form of risk it faces. The standard is structured around 3 axes which define (1) the principles, (2) the process, and (3) the organizational framework of risk management. The risk management process, detailed in ISO 31000 and presented in Figure 2, is an iterative process that encompasses 6 parts: (1) Communication and consultation, (2) Scope, context and criteria, (3) Risk assessment, (4) Risk treatment, (5) Monitoring and review, and (6) Recording and reporting. In our paper, we used the process described in this standard to build our comprehensive risk assessment method, i.e., risk identification, analysis, and evaluation.

2.5. STRIDE

STRIDE is a cybersecurity threat modeling methodology developed by Loren Kohnfelder and Praerit Garg [14] in 2009, and adopted by Microsoft in its Security Development Lifecycle (SDL). Supported by Microsoft, this methodology is well-known among vendors and solution providers. STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege. These threats and the security objectives they violate, were defined by Shostack in 2014 [15] as:

- **Spoofing (Authentication):** Pretending to be something or someone other than yourself.
- **Tampering (Integrity):** Modifying something on a disk, a network, or in memory.
- **Repudiation (Non-repudiation):** Claiming that you did not do something or were not responsible. Repudiation can be honest (truthful claim) or false (a lie).
- **Information disclosure (Confidentiality):** Providing information to someone not authorized to see it.

[2]The standard defines this language as *"deprecated and will not be contained in the next edition of this standard"*

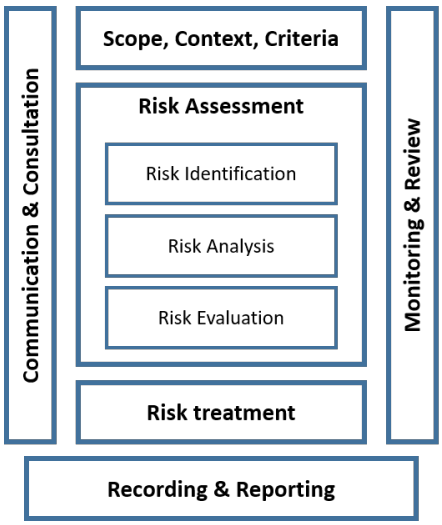

Figure 2: ISO 31000 Risk Management Process

- **Denial of service (Availability):** Absorbing resources needed to provide a service.
- **Elevation of privilege (Authorization):** Allowing someone to do something they are not authorized to do.

In this paper, we use a threat modeling tool based on the STRIDE methodology in order to identify vulnerabilities in the system.

3. Methodology

The proposed methodology automates an ISO 31000 [4] risk assessment process applied to ICS. More precisely, we provide an automated cybersecurity risk assessment for the safety of ICS that can be used to partially automate the initial (step 2) and detailed (step 5) risk assessments of IEC 62443-3-2. Figure 3 details the steps that can be automated by our methodology and specifies which part of our methodology makes this possible.

The IEC 62443-3-2 standard process includes both an initial and a detailed risk assessment, as presented in Section 2.2. The standard clearly defines the steps that need to be achieved in order to carry out a detailed risk assessment (Step 5), which is not the case for the initial risk assessment (Step 2). However, the standard specifies that both the initial and the detailed risk assessment should be derived from the same source (standard, framework, etc.) to produce consistent and coherent results. In addition, the first five steps of a detailed risk assessment and of the initial risk assessment both aim to determine unmitigated cybersecurity risk. Given these observations, we can assume that the initial risk assessment is in line with steps 5.1 to 5.5 of the detailed risk assessment.

In the following, we first introduce an overview of our methodology (Section 3.1). We then detail each part of the methodology and give an extensive description of their automation.

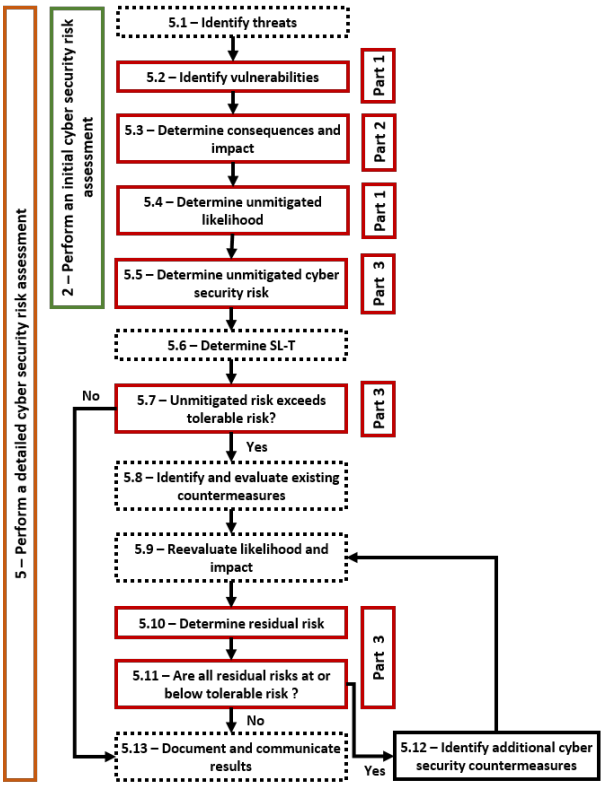

Figure 3: IEC 62443-3-2 risk assessments processes and automation

3.1. Overview

Figure 4 shows an overview of our methodology which comprises 3 parts: (1) threat modeling tool automation, (2) generation of safety-compromising cyberattack scenarios, and (3) risk matrix building.

Part 1. The first part of the method automates a threat modeling tool in order to identify the system's threats and vulnerabilities and to determine their likelihood. From the PLC programs in XML format (PLCOpen format), we extract the PLC communication functions with an XML parser from which we are able to build a Data Flow Diagram (DFD) of the system. Subsequently, this DFD and a threats and vulnerabilities knowledge base informs the threat modeling tool, enabling it to identify the threats and vulnerabilities of the system. Finally, we assign a likelihood for all the system's vulnerabilities. The knowledge base includes both CVE (Common Vulnerabilities and Exposures) and custom vulnerabilities. The method allows us to automatically assign a likelihood for all the system's CVE and, as the custom vulnerabilities likelihood assignment cannot be automated, their likelihood is assigned by a cybersecurity expert.

Part 2. The second part relies on a method we designed to generate safety-compromising cyberattack scenarios. The method analyzes the PLC's control codes in order to detect

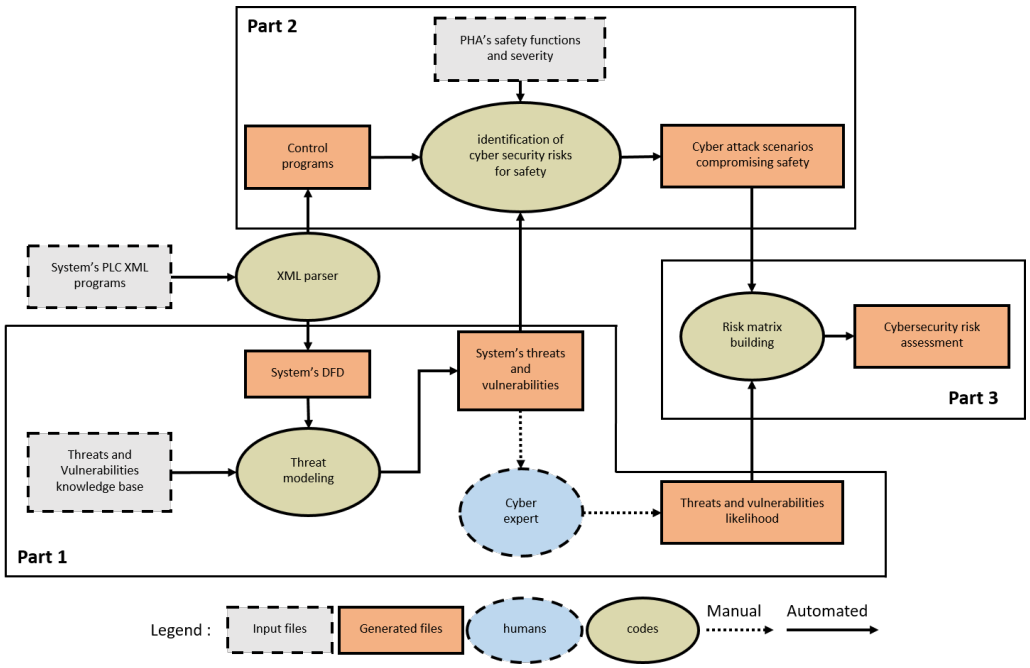

Figure 4: Overview of the Methodology

manipulation of sensor readings and control commands that could compromise the system's safety functions. Safety functions are determined from a process hazard analysis (PHA) and the control codes are extracted from the PLCOpen files (XML) thanks to an XML parser. Finally, the method generates attack scenarios by determining feasible sensor readings and control command manipulation according to the system's threats and vulnerabilities, identified in the first part of our methodology (Part 1).

Part 3. The third part builds a risk matrix that evaluates the cybersecurity risk for the safety of the system. This risk matrix ranks the cyberattack scenarios, generated in Part 2, according to their likelihood and severity. Likelihood of an attack scenario is conditioned by the vulnerabilities likelihood assigned in Part 1, which composes the attack scenario, and severity is determined according to the hazard impact, pursuant to the PHA caused by the safety function compromised by the attack scenario. In a classical PHA, the hazard impact can be assessed according to several criteria such as health, safety, environment, business interruption, financial factors, and so on.

We now take a closer look at each part of the method and the detailed automation of the process parts according to whether they are manual, automated, or automatable. We distinguish between the terms *automated* and *automatable* depending on whether an automation code has been developed.

3.2. Part 1: Threat Modeling Automation

The first part of this methodology uses a graphical threat modeling tool provided by Microsoft, the Microsoft Threat Modeling Tool (MTMT) [16]. The tool works in three phases: (i) template creation, (ii) system modeling, and (iii) model analysis, as depicted in Figure 5.

- **Template creation:** Modeling a system in MTMT requires a *template*, i.e., a library of components and a *knowledge base* of vulnerabilities categorized by the STRIDE threat family (Spoofing, Repudiation, etc.). MTMT proposes three templates by default, available online on Github[3]. These model standard systems (*Default* template), Azure Cloud services (*Azure Cloud Services* template), and medical devices (*MedicalDeviceTemplate*). Each template includes MTMT domain-specific components. For instance, the *MedicalDeviceTemplate* template includes specific communication protocols such as the well-known controller area network (CAN bus).
- **Model building:** In MTMT, the system to be analyzed is modeled as a Data Flow Diagram (DFD), connecting components together with directional arrows. Each generic element of the DFD (process, data store, flow, etc.) is represented by a *stencil* and can be derived to model specific elements (*derived stencil*).
- **Model analysis:** MTMT has a model analysis feature that automatically generates a list of exploitable vulnerabilities. Each exploitable vulnerability in the list is associated with the STRIDE threat it represents.

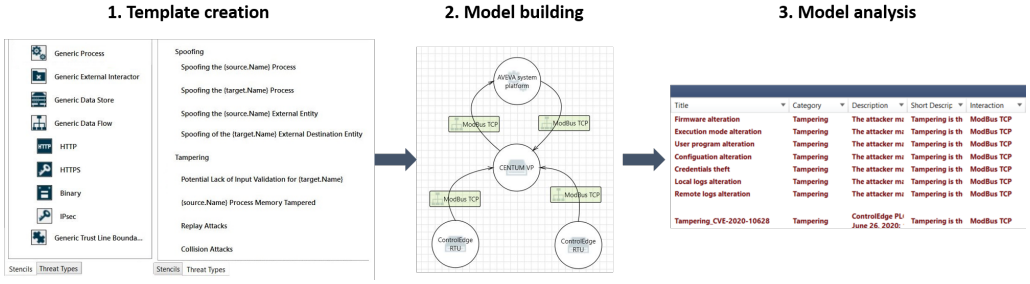

Figure 5: Microsoft Threat Modeling Tool Workflow

3.2.1. ICS Template Automation for MTMT

In a previous paper [5], we presented a toolkit to generate ICS templates for MTMT. Available online[4], this toolkit tackles the lack of ICS components currently modeled in MTMT and allows the threats to this type of system to be correctly modeled. Figure 6 presents the three features of the toolkit developed in [5].

1. An extensible database of ICS components that stores companies' knowledge of industrial devices, industrial dataflows, and their vulnerabilities in order to improve the system's modeling.

[3] `https://github.com/microsoft/threat-modeling-templates?tab=readme-ov-file`
[4] `https://github.com/StrideICS/StrideICS`

2. A method to model known vulnerabilities in MTMT by collecting and filtering known vulnerabilities from a public CVE (common vulnerabilities and exposures) database.
3. A tool that automates translation of the ICS extensible database (feature 1) and CVE modeling (feature 2) into an MTMT template.

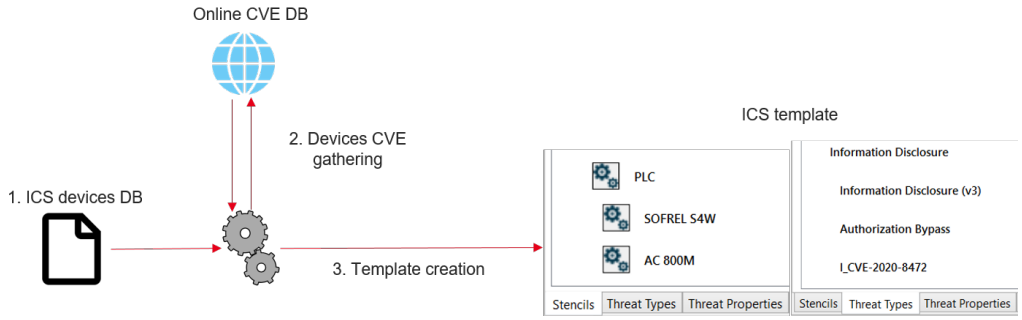

Figure 6: Overview of the Toolkit Developed in [5]

3.2.2. MTMT Model Building

In comparison to [5], we automate model building in this paper using PLC programs to provide automation for each step of the MTMT workflow (Figure 5), in other words, (1) template creation, (2) model building, and (3) model analysis. Indeed, the template creation was automated in [5] as presented in Section 3.2.1, the model building automation process is presented in this section, and finally, the model analysis is already automated by MTMT.

The IEC 61131-3 standard specifies several programming languages to develop logic control and signal/data processing program applications for PLC. One of the programming languages is the function block diagram (FBD) that links function blocks together with signal flow lines to form a diagram. The IEC 61131-5 standard extends the scope of IEC 61131-3 programs by defining function blocks able to describe the PLC communication functions. We use this description of PLC communication functions to build the system's MTMT model.

The PLC communication standard provides the control system with eight communication functions [13], 6 of which have function blocks specified in the IEC 61131-5 standard. In this paper, we examine the data exchange functions presented in Table 1, with their corresponding function block. The connection management, which is used by the other blocks, is provided by the *CONNECT* function block that creates a *COM_CHANNEL* variable to identify the remote partner of a data exchange. These communication functions blocks, instanced in PLC programs, contain all the information needed to build the threat modeling tool DFD.

A DFD is generally built by linking system entities with dataflows. To this end, for each entity, we need to identify which dataflow is transmitted and to where. Each communication function block listed in Table 1 has a parameter named *ID* that determines the remote communication partner (variable provided by the *CONNECT* function block), allowing us to determine the equipment that communicate together. Three parameters

Communication function	Communication function block name
(Polled) Data acquisition	READ
(Programmed) Data acquisition	USEND & URCV, BSEND & BRCV
(Parametric) Control	WRITE
(Interlocked) Control	SEND & RCV
(Programmed) Alarm reporting	NOTIFY, ALARM

Table 1: PLC Communication Functions used for Variable Exchange and their Corresponding Function Blocks

define exchanged data: *SD_ i*, which defines the data to send; *RD_ i*, which identifies the last data received; and *VAR_ i*, which identifies a remote partner variable. All the function blocks have at least one of these parameters. For instance, the *READ* function block contains parameters *VAR_ 1* to *VAR_ n* as inputs that define the remote variables to read and parameters *RD_ 1* to *RD_ n* as outputs that define the last value reading of the remote variables. In addition, the function block specification allows us to determine the flows that the *READ* function block generates. The variables and the corresponding dataflow that handle it are stored in a file that we will use later in Section 3.3. Table 2 shows dataflows generated by the function blocks and the data they handle according to their specifications and their parameters *SD_ i*, *RD_ i*, and *VAR_ i*.

However, an MTMT model represents flows between entities (system devices) called stencils (or derived stencils). Thus, to build the model, we need to map each device with its corresponding stencil (or derived stencil), the function blocks it executes, and its channel communication identifier (the ID parameter that identifies the device as the remote communication partner). This information is provided through an association file, as follows:

- **Device Name**: A unique name provided by the modeler to identify the device in the DFD
- **Stencil or derived stencil name**: The stencil or derived stencil name, as defined in the MTMT template (the library of components) of the corresponding device type.
- **Function blocks executed**: the PLCOpen file path. As noted in Section 2.3, the IEC 61131-10 (PLCOpen format) provides a standard format for exchanging XML-based 61131-3 programs. As the PLC communication functions are described in a 61131-3 function block diagram programming language, we can retrieve all the information about the function blocks executed by the device from this XML file.
- **Channel communication identifier**: The *COMM_ CHANNEL* variable used to identify the device (the ID parameter of the function blocks)

In an MTMT model, dataflows are also represented by stencils that define the communication protocol used. Generally, the function blocks detailed in the IEC 61131-5 are developed by manufacturers for each protocol with a unique function block name. Thus,

Communication function	Communication function block data flow
(Polled) Data acquisition	1.READ $\xrightarrow{\text{VAR\_1:VAR\_n}}$ Remote partner 2.READ $\xleftarrow{\text{RD\_1:RD\_n}}$ Remote partner
(Programmed) Data acquisition	USEND $\xrightarrow{\text{SD\_1:SD\_n}}$ URCV, BSEND $\xrightarrow{\text{SD\_1}}$ BRCV
(Parametric) Control	WRITE $\xrightarrow{\text{(VAR\_1, SD\_1):(VAR\_n, SD\_n)}}$ Remote partner
(Interlocked) Control	1.SEND $\xrightarrow{SD_{SEND}\_1 : SD_{SEND}\_n}$ RCV 2.SEND $\xleftarrow{SD_{RCV}\_1 : SD_{RCV}\_n}$ RCV
(Programmed) Alarm reporting	NOTIFY $\xrightarrow{\text{SD\_1:SD\_n}}$ Remote partner, ALARM $\xrightarrow{\text{SD\_1:SD\_n}}$ Remote partner

Table 2: Function Block Dataflows for Variable Exchange

we can define the stencil (representing the dataflow) of a function block according to its name.

Example. To illustrate the building process of an MTMT model, we detail an example of a function block for communication between two PLCs. To build the model, we first create the association file as follows:

- PLC1 (stencil: PLC, PLCOpen: plc.xml, ID: 192.168.1.1)
- PLC2 (stencil: PLC, PLCOpen: None, ID: 192.168.1.2)

As an example, we detail a Schneider Electric implementation of a *BSEND* function block for a user datagram protocol (UDP) instanced in the PLC1 XML file. This function block contains the following parameters:

- *i_sPeerIP*: Message destination address;
- *i_uiPeerPort*: Message destination port;
- *i_pbySendBuffer*: Start address of the buffer containing the data to be sent.
- *i_udiNumBytesToSend*: Number of bytes to be sent from the buffer provided by the application.

We retrieve the corresponding values of these parameters for the *USEND* instance in the XML file:

- *i_sPeerIP* is equal to **192.168.1.2** (PLC2),
- *i_uiPeerPort* is equal to **2000**,
- *i_pbySendBuffer* is equal to **16-bits Memory address 225** (%MW225),
- *i_udiNumBytesToSend* is equal to **1**,

This function block sends data stored at the memory address 225 of the PLC1 to the PLC2 (192.168.1.2). Moreover, as the XML file contains the control code, we can

determine the corresponding variable names of the values sent. From this function block, we can define two PLC stencils in the DFD as well as one UDP dataflow from PLC1 to PLC2, as defined in Table 2. In addition, we store the match between the dataflow generated and the variables exchanged in a file.

3.2.3. Likelihood Assignment

The Microsoft Threat Modeling Tool (MTMT) proposes a model analysis function that automatically generates a list of the system's vulnerabilities from a knowledge base. We extended this list with a likelihood assignment for each exploitable vulnerability extracted by the tool from the knowledge base. Likelihood is determined according to the common vulnerability scoring system (CVSS) V3.1 [17] exploitability score. The scoring system has the advantage of being harmonized and widely used. CVSS version 4 was released in 2023, and includes additional applicability to OT, ICS, and IoT [18], but the CVE database that we use, the NIST's National Vulnerability Database (NVD), still uses CVSS version 3.1 which appears to remain the default rating system.

The vulnerabilities knowledge base is composed of known vulnerabilities gathered from the common vulnerabilities and exposures (CVE) database and from custom vulnerabilities such as a lack of authentication mechanism. The CVSSCVE v3.1 score has already been calculated for CVE, so we can automatically assign their likelihood. The vulnerability described in CVE-2023-1150[5], for instance, has an exploitability score of 3.9 (the maximum score). Custom vulnerabilities do not provide enough information to automatically compute their exploitability score and require a cybersecurity expert to do this. The exploitability score is calculated according to the equation (1) provided by the CVSS v3.1 specification [17], which involves four metrics: the Attack Vector (AV), the Attack Complexity (AC), the Required Privilege (RP), and the User Interaction (UI). Table 3 details the scoring system for each exploitability metric.

$$Exploitability = 8.22 \times AV \times AC \times PR \times UI \tag{1}$$

3.2.4. Automation

Part 1 aims to automate identification of the system's vulnerabilities and the likelihood assignment process through a threat modeling tool. Figure 7 shows the process that automates the MTMT workflow (cf. Figure 5), i.e., (1) template creation, (2) model building, and (3) model analysis.

- **Template creation:** The template creation is achieved via a toolkit that we developed and and made freely available online[6].
- **Model building:** The model building, described in Section 3.2.2, requires a user to create the association file and a cybersecurity expert to check the completeness of the built DFD. Otherwise, the process can be automated.
- **Model analysis:** The model analysis, already automated in MTMT, generates a list of threats and vulnerabilities that is exportable as a CVS file.
- **Likelihood assignment:** We automate the likelihood assignment for CVE. Custom vulnerabilities need a cybersecurity expert to complete their likelihood assignment.

[5] `https://nvd.nist.gov/vuln/detail/CVE-2023-1150`
[6] https://github.com/StrideICS/StrideICS

Metric	Metric Value	Numerical Value
Attack Vector (AV)	Network	0.85
	Adjacent	0.62
	Local	0.55
	Physical	0.2
Attack Complexity (AC)	Low	0.77
	High	0.44
Privileges Required (PR)	None	0.85
	Low	0.62
	High	0.27
User Interaction (UI)	None	0.85
	Required	0.62

Table 3: CVSS v3.1 exploitability scoring system

In the first part, we saw how the MTMT model is built from the PLC communication function blocks to automate the threat modeling process, and how we assign a likelihood to the vulnerabilities identified. In the next part, we detail how we identify vulnerabilities with an impact on the system safety.

3.3. Part 2: Finding Consequences and Impacts from the PLC logic

In this section, we first present an earlier paper [6] that determined the consequences and impact of cyberthreats for the system's safety. We then explain in detail how we provide the input data for the [6] method and, more importantly, how we integrate the outputs of the threat modeling tool.

In an earlier paper [6], we proposed an integrated methodology to identify cybersecurity risks through attack scenarios that compromise system safety. These scenarios involve manipulation of sensor readings and control commands to compromise the safety functions provided by the system. The approach is summarized in Figure 8 and follows three sequential steps. In the first step, the process is modeled by converting the PLC program into a finite automaton. We added several improvements to the PLC program's conversion into the finite automaton from the original work [6]. These are presented in Section 4. Within this automaton, we then identified the safety functions that the system must guarantee, and finally, attack scenarios are generated that compromise the safety functions for system threats. This methodology requires three forms of input: i.e., all the PLC programs for the modeling step, the safety functions that the system must guarantee, and the system threats. The rest of this section details how we add the three inputs to the methodology.

3.3.1. Retrieving PLC Programs

The PLCOpen format proposes an XML-based file for IEC 61131-3 programs. IEC 61131-3 specifies several programming languages to develop logic control and signal/data processing program applications for PLCs. The first step is to export every PLC IEC 61131-3 program into PLCOpen format.

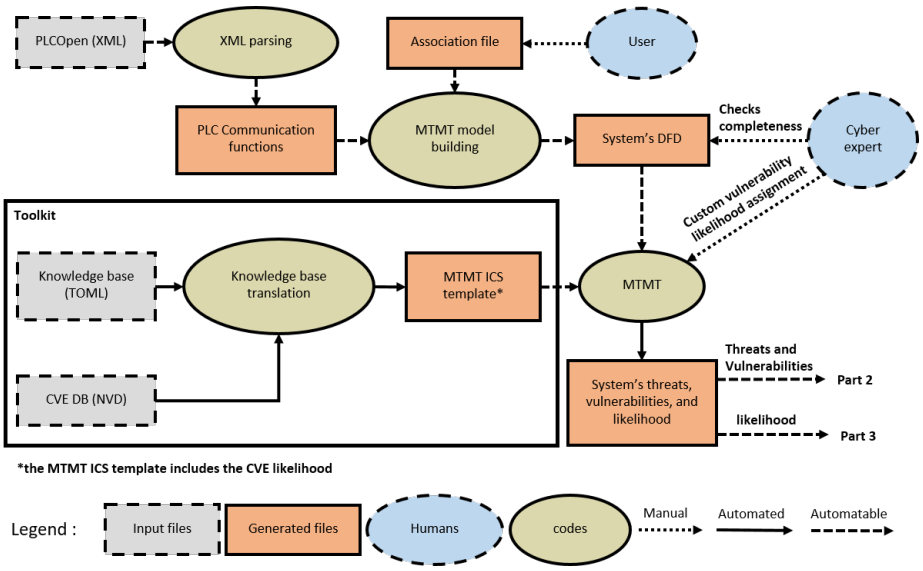

Figure 7: Part 1 Process and Automation

3.3.2. Retrieving System Safety Functions

The first step in our methodology is a process and hazard analysis (PHA), as specified by the IEC 62443-3-2 standard. PHA aims to identify both the hazards of a process and an assessment of the associated risk. Generally, the risk is assessed according to a risk matrix linking the likelihood of the event and the severity of its consequences in the system.

Schematically, an industrial control system is a set of inputs that determine the process state (sensors and PLC internal variables) and a set of outputs that determine the command state. A safety function defines the connection between a process state and a command state, allowing the system to be protected from hazard. For instance, in

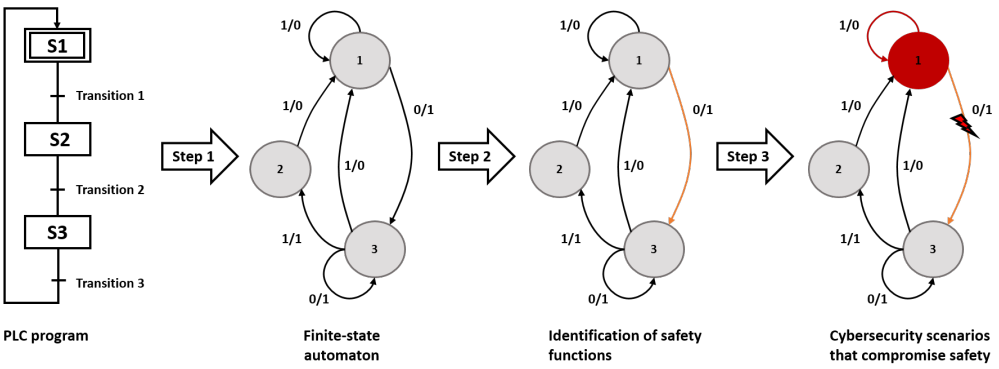

Figure 8: Safety-Security Risks Identification Method Overview

order to avoid a tank overflow, if the tank is full, then the filling valves must be closed. This **IF ELSE** structure is a *material implication*, $A \implies B$ whose truth table is $\neg A \vee B$. The methodology developed in [6] uses this formalism to describe the safety functions.

We derived the material implications from the hazards the system has to face (identified by the PHA). Indeed, a hazard is a system state, i.e., a combination of a process state and a command state, which represents a risk for assets, people, and the environment. The safety function must prevent this state.

IF

$$Hazard = processState \wedge commandState$$

Then

$$Safety = \neg(processState \wedge commandState)$$

$$Safety = \neg processSate \vee \neg commandState$$

$$Safety = processState \implies \neg commandState$$

We express material implication antecedents (the left part) by PLC inputs and consequent (the right part) by PLC outputs, as this corresponds to the PLC's execution cycle (input readings, program execution, output updates), where inputs imply outputs.

3.3.3. Retrieving Threats to the System

In [6], two different types of attack scenario are defined, namely, upstream and downstream attacks, according to whether they occur before or after a safety event. They are mapped according to four possible threats in the industrial network: (i) communication interruption, (ii) packet forging, (iii) information reading[7], and (iv) information modification that enables data manipulation to compromise safety. We also detailed in [6] that upstream attacks aim to mask data variables refreshment and require a denial of or tampering with data variables, while downstream attacks aim to force an output by injecting data variables, and involve tampering with or forging data variables.

An attack represents the realization of a threat and requires the exploitation of one or several vulnerabilities. Thus, we identified feasible threats (communication interruption, packet forging, etc.) by mapping these threats with the system's vulnerabilities extracted by the threat modeling tool. We achieved this mapping according to the common security properties violated by the threats and vulnerabilities (categorized into STRIDE threat families) as defined in Table 4.

Threat	**Threat (STRIDE)**	**Violated security property**
Communication interruption	Denial of service	Availability
Packet forging	Spoofing (source)	(source) Authentication
Information modification	Tampering	Integrity

Table 4: Mapping Vulnerabilities and Threats

[7] As a side note, even if an information reading can be used to prepare other attacks, we consider it negligible as it has no direct impact on safety.

We assume that the variables contained in a vulnerable channel are also vulnerable, i.e., if a dataflow is vulnerable to tampering, we consider that the variables handled by this dataflow are also falsifiable. As a reminder, in Section 3.2.2, we created a file that mapped dataflows with the data they handle. This allowed us to determine potential data manipulation, enabled by attacks, to build attack scenarios.

Example. Figure 9 shows a Microsoft Threat Modeling Tool (MTMT) model of a control loop involving a Modbus TCP communication between a WAGO 750-3x/-8x series PLC and a remote terminal unit (RTU). In this example, the RTU acts as an analog-to-digital converter for sensor values and as a digital-to-analog converter for control commands. The PLC executes a control code that trips the circuit breaker in case of electrical hazard.

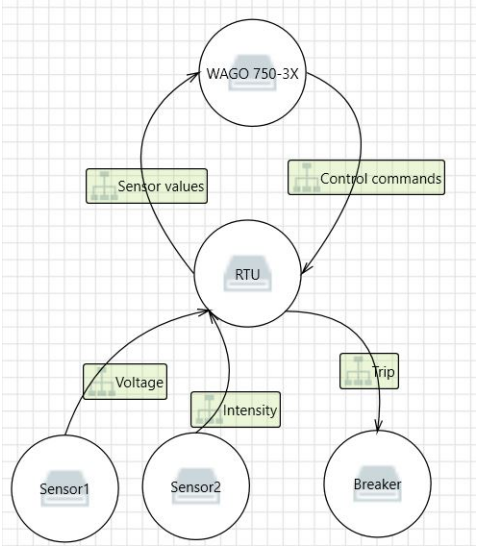

Figure 9: Example of an MTMT Control Loop Model

Our PLC logic-based method identifies a possible upstream attack scenario against the system by masking an overcurrent on the power grid in order to avoid tripping the circuit breaker. To succeed in this upstream attack, an attacker can interrupt or alter information from the communication channel between the PLC and the RTU (masking data refreshment). Table 4 shows that to interrupt or alter communication information, an attacker needs to exploit, respectively, a vulnerability that realizes a *Denial of Service* or a *Tampering* STRIDE threat.

On the other hand, the MTMT model analysis feature extracts an exploitable vulnerability from the system's model (Figure 9), detailed in CVE-2023-1150[8], which allows an unauthenticated remote attacker to deny the Modbus server service to WAGO 750-3x/-8x products. Thus, we can generate an upstream attack scenario where an unauthenticated remote attacker denies the PLC's Modbus server service (threat) by sending specially crafted packets (vulnerability CVE-2023-1150) to mask an overcurrent (upstream attack) and so prevent the circuit breaker from tripping (compromising the safety function).

[8] `https://nvd.nist.gov/vuln/detail/CVE-2023-1150`

3.3.4. Automation

This second part automates our method to identify cybersecurity risk for the safety of the system. The process beginning by the PHA in Figure 10 requires a safety expert to format the system's hazards (coming from the PHA) into a material implication. Each material implication is associated with its severity risk evaluation (determined by the PHA). The material implication is the format we use to model safety functions. Thus, we can derive the system's safety functions from the hazards formatted as material implications, as explained in Section 3.3.2, and their corresponding severity in the case of compromise. Section 3.3.3 details how to match vulnerabilities from MTMT (Part 1 of the method) with the threats used to generate attack scenarios. Finally, we developed a tool that automates the process modeling, described later in detail in Section 4.2.1, and the generation of attack scenarios.

- **TELECO tool:** TELOCO is a tool provided by Provost et al. [19] that automates SLA modeling from an SFC code. The code is no longer available at the link provided in Provost et al. [19], but a Github project is available with the code at [20].
- **Sub-process modeling:** The tool we developed includes sub-process modeling. As presented later in Section 4.2.1, we group together SLA that manage a distributed safety function using a strong product of graphs.
- **Attack scenarios:** Attack scenarios are generated according to whether they occur before (upstream attacks) or after (downstream attacks) an SLA transition corresponding to execution of the safety function. Our code implements identification of safety functions into the sub-processes and building of upstream and downstream attacks.

To summarize, we determined the consequences of cyberattacks on the system's safety functions (Part 3) by mapping the vulnerabilities identified through a threat modeling process (Part 2 - Section 3.2), adopting a methodology that generates safety-compromising attack scenarios. In the last part, we evaluate the level of risk of the attack scenarios generated.

3.4. Part 3: Risk Evaluation

The final step generates a risk matrix according to the likelihood and impact of risk. Section 3.2.3 assigns an exploitability likelihood to the system's vulnerabilities and Section 3.3 links these vulnerabilities to attack scenarios that compromise the safety of the system. We thus determine the likelihood of a risk (attack scenario) according to the exploitability likelihood of vulnerabilities that compose the attack scenario and the impact of the risk according to the hazard severity value associated with the safety function compromised by the attack scenario.

More specifically, we establish the likelihood rating scale based on the CVSS v3.1 exploitability score that ranges from 0.1 to 3.9, and which we map to the CVSS v3.1 qualitative severity rating scale using a cross product: $Low = 0.1 - 3.9$, $Medium = 4.0 - 6.9$, $High = 7.0 - 8.9$, $Critical = 9.0 - 10.0$. We do not include the *None* rating (score equals zero) as the exploitability score is necessarily greater than zero, depending on its equation (cf. equation 1). The impact is determined by the severity rating scale used by the process and hazard analysis (PHA). In a classical PHA, the impact can be assessed based on several criteria: health, safety, environment, business interruption,

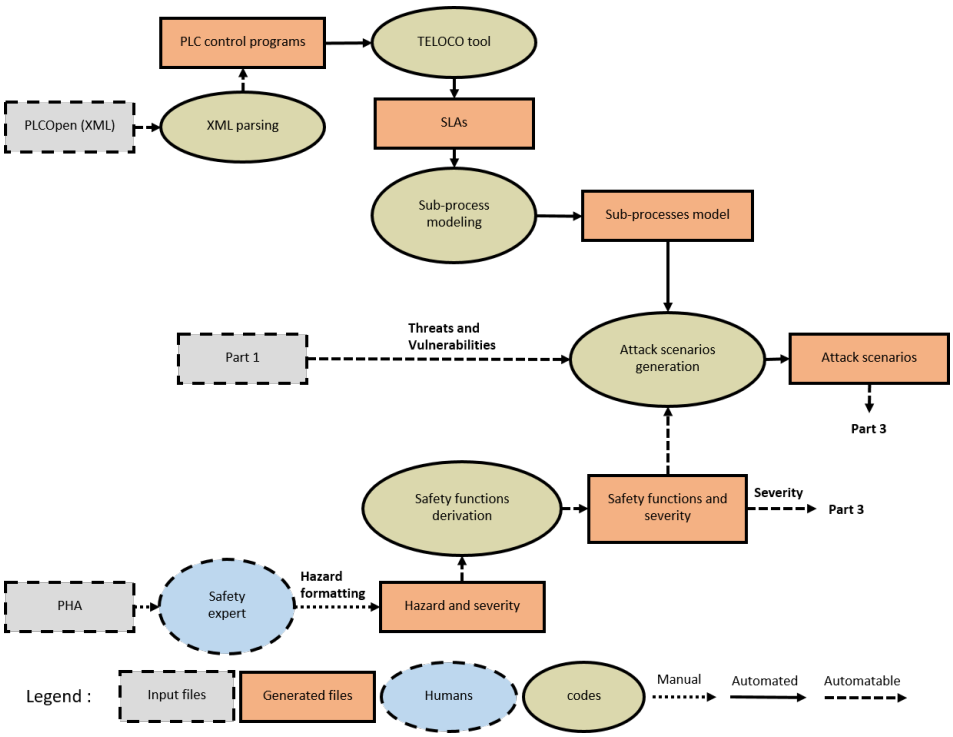

Figure 10: Part 2 process and automation

financial factors, etc. that each use their own metrics, such as cost in dollars or loss of physical integrity.

Finally, we evaluate the risk by identifying matrix cells (a likelihood/impact pair) which correspond to a level of risk that is not tolerable for the organization. Consequently, none of the attack scenarios included in these cells are acceptable to the organization and thus require additional treatment.

3.4.1. Automation

The process in the third part, depicted in Figure 11, is intended to automate the risk evaluation, i.e., to compare the risk analysis results with the tolerable risk for the organization. Usually a risk analysis determines the level of risk by combining the likelihood and severity of risks. In our method, cybersecurity risk is identified through the attack scenarios generated in Part 2. We determine the level of risk of the attack scenarios according to an exploitability score of the vulnerabilities that go into an attack scenario (likelihood) and the hazard severity caused by the safety function compromised by the attack scenario (Part 2). The attack scenarios' level of risk allows them to be ranked in a risk matrix. The risk matrix can be built automatically according to the severity scale used by the PHA and the likelihood scale used by the CVSS v3.1 (cf. Section 3.4).

Finally, the risk evaluation may be automated by highlighting the organization's tolerable risk in the matrix through a match between a cell color and its level of risk. For

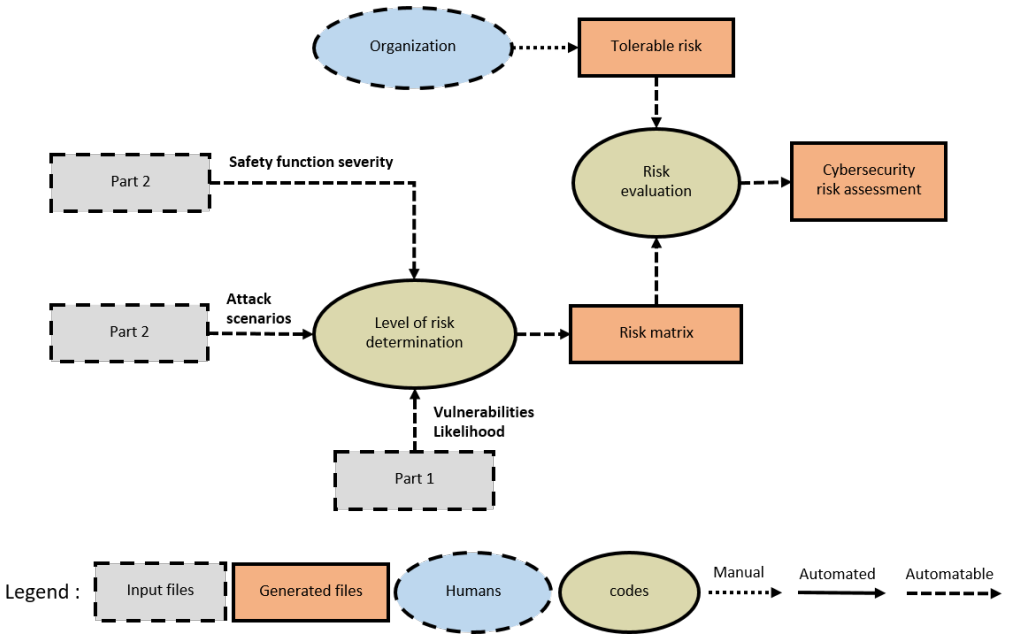

Figure 11: Part 3 process and automation

example, matrix cells which represent a risk that is not tolerable could be distinguished by a red color.

This part of the method can be used to automate the risk evaluation steps 5.5, 5.7, 5.10, and 5.11 of the detailed risk assessment (cf. Figure 3). Indeed, step 5.5 aims to generate a risk matrix based on the likelihood and severity scores of attack scenarios, and step 5.10 aims to generate new risk matrices with the likelihood and severity scores mitigated by additional cybersecurity countermeasures (steps 5.12 and 5.9). Steps 5.7 and 5.10 respectively compare whether the unmitigated risk and the mitigated risk exceed the tolerable risk, i.e., in our case, whether the attack scenarios belong to a risk matrix cell that corresponds to a non-tolerable risk for the organization.

In this section, we presented our risk evaluation process which classifies attack scenarios pursuant to their level of risk and assesses whether they meet the organization's tolerance threshold.

4. Evaluation

This section evaluates our safety-security risk assessment in accordance with two criteria: comprehensiveness and scalability to large systems.

4.1. Comprehensiveness

In the literature, the term *risk assessment* is defined differently, depending on the study. Our approach assesses the methods' completeness in line with ISO 31000, which defines risk assessment as a three-stage process, contributing to the identification, analysis, and evaluation of risk.

Risk identification. Risk identification aims to identify, recognize, and describe the risk. In Section 3.3, we presented a methodology we designed to identify cybersecurity risk for a system's safety. This method includes the identification of 3 threats (communication interruption, packet forging, and information modification) that could compromise the safety of a system by means of attack scenarios.

Risk analysis. Risk analysis aims to understand the nature and level of risk so as to provide the data required for risk assessment. The level of risk is usually given by the likelihood and severity of a risk. Our method automates a threat modeling tool that identifies system vulnerabilities according to the existing cybersecurity controls. We then automatically assign an exploitability score to each vulnerability, except for custom vulnerabilities. Afterwards, this score is used to compute the likelihood of attack scenarios, in other words, the likelihood of the cybersecurity risk for the system's safety. Our attack scenarios compromise the system's safety functions. We use the hazard caused by these compromised safety functions to assign a severity score to the attack scenarios. Finally, depending on the likelihood and severity score, we set a level of risk to each attack scenario.

Risk evaluation. Risk evaluation compares the level of risk with that tolerated by the organization. To this end, we developed a risk matrix that ranks each attack scenario in accordance with its level of risk (likelihood and severity score). To evaluate the risk, the organization sets the matrix cells, the combination of non-tolerable likelihood and severity scores, while simultaneously identifying attack scenarios that require additional treatment to meet the tolerable risk threshold.

4.2. Scalability

The main bottleneck in safety-security risk assessment is the balance between the level of granularity, i.e., the amount of detail included in the model, and the system's complexity. As we work with safety and security, we aim to provide both a highly detailed cyberattack description to implement the more relevant counter-measures and a precise description of the process to clearly identify the consequences on safety. However, a high level of cybersecurity and safety implies an unacceptable safety-security state space that generates combinatorial explosion. Our work offers a balance between safety and security modeling in order to scale up to large use cases. To evaluate the scalability of our method to large systems, we performed a complexity analysis for our process modeling (Section 4.2.1) and our attack scenario search algorithm (Section 4.2.2).

4.2.1. Process Modeling Complexity

The process modeling analyzed in this section is an improved model of [6]. Below, we outline the main changes and their impact on the process modeling complexity. As a remainder, we model the process by translating the PLC logic (control codes) into one or more finite state automata. These automata clearly describe the process states and events managed by the control system, which are implicitly specified in the IEC 61131-3 programming languages.

PLC logic modeling into finite state automaton. The first improvement is the use of Stable Location Automaton (SLA) instead of a Mealy machine to model the PLC Logic. Provost et al. [19] propose a GRAFCET (specification format close to the SFC syntax) to Mealy machine translation for conformance test purposes through an intermediate model called Stable Location Automaton (SLA).

In our earlier work [6], we used this method to model the SFC programs into Mealy machines. As discussed in [6], despite a few differences between the GRAFCET and the SFC, we can consider that their operation is equivalent in our work. Provost et al. [19] provided a tool called TELOCO, available online[9], to perform the method automatically. In the method [19], the built Mealy machine is specified in full, i.e., all the Boolean input combinations are explicitly expressed for each state through transitions ($|transitions| = 2^{|inputs|} \times |states|$). For instance, for a Mealy machine with 2 inputs (A and B) and 2 states ($S1$ and $S2$), the fully specified Mealy machine includes 8 transitions, 4 transitions ($2^{|inputs|}$) for state $S1$ and 4 transitions for state $S2$, as presented in Figure 12. To improve the legibility of the figure, we omit the Mealy machine outputs and differentiate transitions arising from $S1$ (dashed lines) and $S2$ (solid lines).

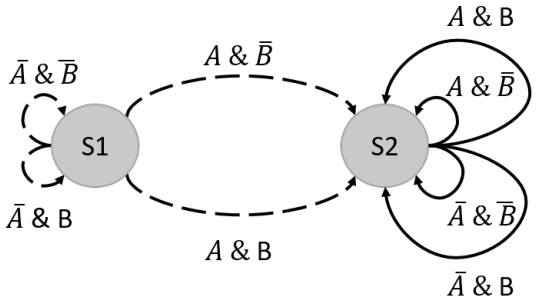

Figure 12: Example of a Fully Specified Mealy Machine

In our previous work, we were interested in grouping transitions with the same origin and destination state into a single transition by minimizing their Boolean function. For example, in Figure 12, there are two transitions that loop over $S1$ ($\neg A \wedge \neg B$ and $\neg A \wedge B$). The minimal Boolean expression from these two transition is equal to $\neg A$. Consequently, we group both transitions into a single one using the Boolean expression $\neg A$. Figure 13 provides the equivalent Mealy machine with minimized transitions of Figure 12. To compute this Boolean minimization, we used the Quine-McCluskey (QMC) algorithm for each group of transitions. This modeling process has a rapid combinatorial explosion due to the Quine-McCluskey algorithm that has exponential complexity in computation time with the number of input variables.

In the current method, we use the SLA, the intermediate model between a SFC and a Mealy machine. Noting that the Mealy machine with minimized transitions is a specific SLA, we improved the previous modeling process by removing the SLA-Mealy machine translation and the Boolean minimization steps, as these steps are equivalent to stopping the method after buiding the SLA. This improvement is summarized in Figure 14.

[9]The link provided in Provost et al. [19] is obsolete but a fork of the code is available at [20]

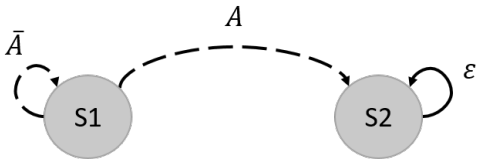

Figure 13: Example of Mealy Machine with Minimized Transitions

Figure 14: SFC to SLA Steps

Construction of an SLA involves computing the set of simultaneous executable transitions, starting from the initial state of the SFC, to discover new states. For these, we again compute the set of simultaneous executable transitions to discover new states until all of them have been discovered. Checking that two transitions are simultaneously executable implies linear complexity depending on the size of the inputs. However, this computation must be done for each combination of executable transitions from the current state. Checking must thus be done $2^{|executableTransitions|}$ times in the worst scenario. In addition, the parallel execution of SFC increases the number of simultaneously executable transitions at least by one. Indeed, two SFC implies at least 2 simultaneously executable transitions (one for each SFC), the parallel execution of three SFC implies at least 3 simultaneously executable transitions, and so on. Thus, the parallel execution of SFC grows exponentially ($2^{|executableTransitions|}$). Moreover, the exponential number of checks also evolves according to the size of the state space, which also grows exponentially for parallel SFC. Executing the SFC in parallel therefore involves a double exponential form of the number of checks to be performed. To resolve this issue, we used the strong product of graphs in order to break the double exponential form.

Strong product of graphs. A strong product of two graphs G and H is a graph such that the set of vertices is the Cartesian product of G and H vertices, and two distinct vertices (ug, uh) and (vg, vh) are adjacent if and only if (a graphical example is provided in Figure 15):

1. $ug = vg$ and uh and vh are adjacent (a transition is only fired in the SFC H), or
2. $uh = vh$ and ug and vg are adjacent (a transition is only fired in the SFC G), or
3. ug is adjacent to vg and uh is adjacent to vh (a transition is fired in both the SFC G and H)

This new graph captures transitions fired only in the first graph (rule 2), only in the second graph (rule 1), and in both graphs simultaneously (rule 3), just as an SLA could do for all the system's SFC. We add another rule to the previous three which involves checking whether the new transitions are compatible. Indeed, two PLC programs can share a same variable and need to have a compatible value, i.e., the variable does not

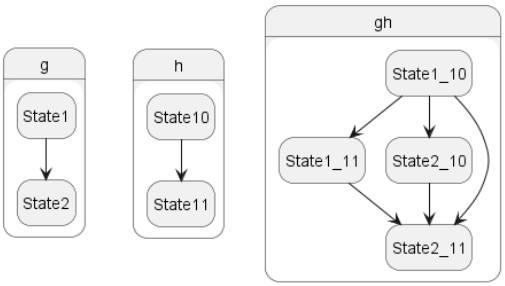

Figure 15: Example of a Strong Product of Graphs

have an opposite value on the new transition. For instance, from Figure 15, if the two transitions (those that connect $State1$ to $State2$ and those that connect $State10$ to $State11$) share a variable with opposite values, we will not be able to fire both transitions at the same time, and thus do not model this option into the final graph as presented in Figure 16.

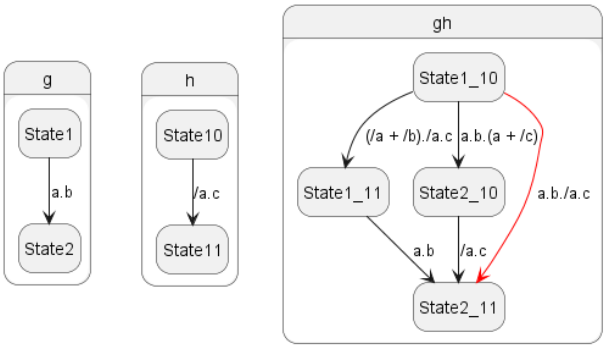

Figure 16: Incompatible Transitions

The strong product of graphs produces a new graph with a state space equal to the product of the state spaces of the original graphs and, in the case of a fully defined SLA, a number of transitions equal to the square of the state space size. For instance, a strong product of two graphs H and G generates a new graph including $|states_G| \times |states_H|$ states and $(|states_G| \times |states_H|)^2$ transitions in the worst case scenario. As parallel graph execution is less complex to model with the strong product of graphs (exponential) rather than building the SLA directly (double exponential), we propose modeling each SFC executed in parallel into an SLA and then computing the strong product of the graphs (SLA), as depicted in Figure 17. However this improvement remains limited as the strong product of graphs still evolves exponentially. For instance, for three graphs including respectively 6, 7, and 8 states, the final graph will include up to 336 states and 112 896 transitions. Therefore, we propose a last improvement to limit the state space of the SFCs executed in parallel, which consists of dividing the system into sub-processes.

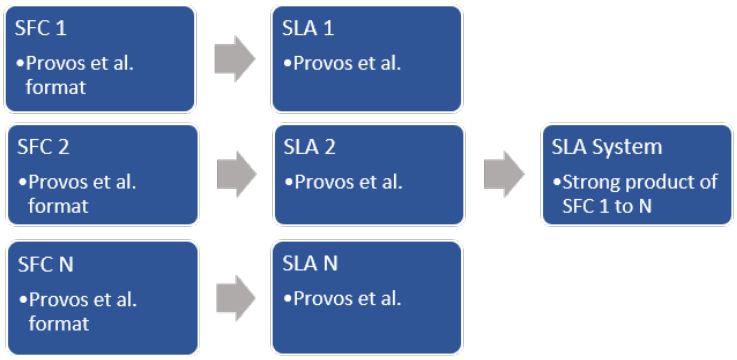

Figure 17: Strong Product of SLA

Division of the system into sub-processes. Our method is designed to identify cyberattacks that compromise the system's safety. Thus, we model two or more PLC logics together only if they implement at least one distributed safety function and we model the process into several sub-processes that provide one or more distributed safety functions. In other words, if the safety function is provided by a single SFC, we translate the SFC into a single SLA. Otherwise, we compute the product space states of SLA that implement the distributed safety function. This decreases the state spaces of the graphs (SLA) and allows us to model the process through a set of sub-processes that autonomously provide the system with a set of safety functions.

Conclusion. This analysis determined the complexity limits of our process modeling and shows that task parallelization severely limits the scalability of our method.

From this observation, we put forward several improvements that decrease the model parallelism and limit the state space explosion in order to model large use cases. Figure 18 describes the Tennessee Eastman chemical process [21], considered as a classic benchmark for industrial system modeling and simulation. Thus, in Figure 18, sensors and actuators of the same color belong to the same SFC, and sub-processes are represented by colored rectangles.

Table 5 presents the time needed to model the blue sub-process of Figure 18 with each of our improvements. Models were created with a laptop that had an Intel(R) Core(TM)i5-8365U @1.60GHz-1.90GHz processor and 16 GB of RAM.

Method	Blue sub-process modeling time	Ratio
Mealy with minimized transitions	1 970 000 ms (32 min 50 s)	9 800
Strong product of Mealy with minimized transitions	24 000 ms (24 s)	119
SLA	1 855 ms (1 s 855 ms)	9
Strong product of SLA	201 ms	1

Table 5: Modeling Time for the Blue Sub-process According to the Improvements

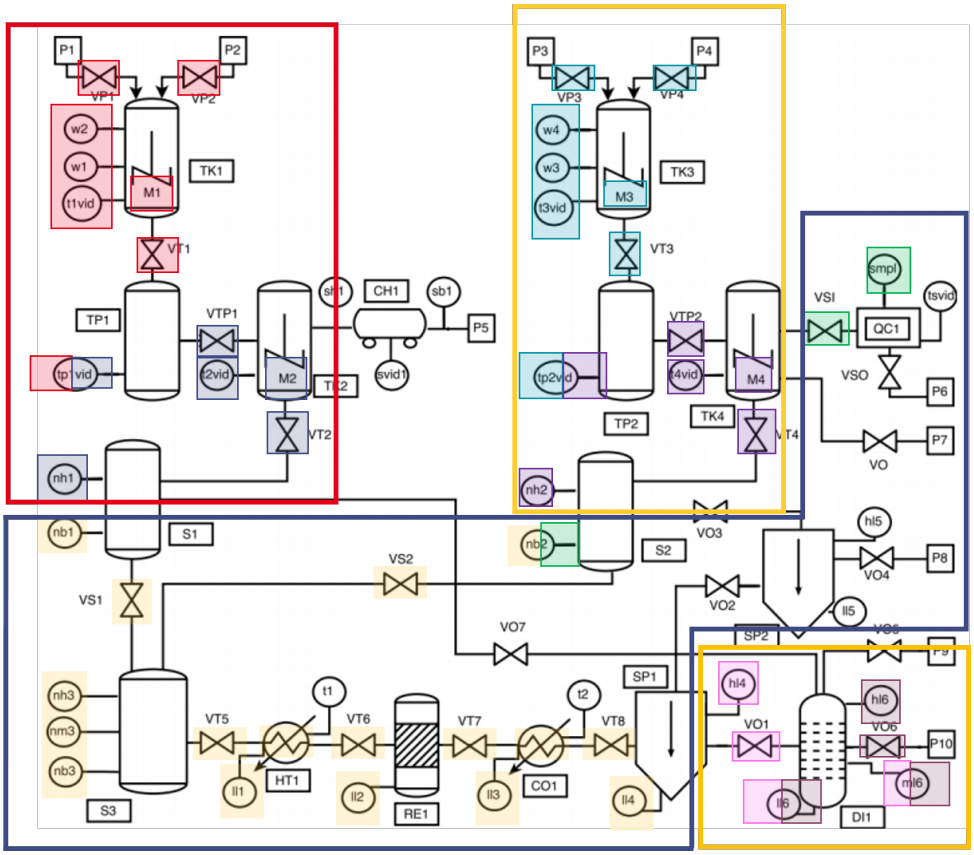

Figure 18: Division of the Tennessee Eastman Chemical Process by Sub-processes

4.2.2. Attack Scenario Search Algorithm

We define two types of data manipulation (attack scenarios), depending on whether they occur before or after an SLA transition that corresponds to the execution of the safety function. Consequently, we need to identify the SLA transitions corresponding to the execution of the safety function to determine the attack scenarios. In the SLA, the safety function execution is represented by a transition between 2 states. Indeed, in addition to the source state and the destination state, an SLA transition includes the event that triggers the change of state and the control command emitted (the new output values). The pair event/command corresponds to the execution of the safety function, and the event implies the command. Thus, we search for each transition that matches the safety functions within the SLA. For example, if we define a safety function $A \implies B$, we iterate over the SLA transitions to find each transition, including A as event and B as command. We say that these transitions execute the safety function and are used to build the upstream and downstream attacks.

The algorithm used to identify safety functions in the model needs to compare the safety function with each transition of the SLA. Therefore, the algorithm complexity

evolves linearly according to the number of transitions. We can check several safety functions at the same time to avoid repeated iterations over the SLA transitions.

In this section, we evaluated our risk assessment method in terms of its completeness and scalability for large systems. We showed that our method is comprehensive and can be applied to large systems such as the Tennessee Eastman chemical process, considered a classic benchmark. We now present the state-of-the-art in safety and security risk assessment, as well as the state-of-the-art in automation of the IEC 62443-3-2 risk assessment in order to compare our method.

5. Comparison with the State-of-the-art

In this section, we compare our work with the state-of-the-art in integrated safety-security risk assessment and the state-of-the-art in automation of the IEC 62443-3-2 risk assessment.

5.1. Integrated Methods

We developed an integrated safety-security risk assessment for ICS. More specifically, we investigated the conditional dependency [22] between safety and security, where system safety depends on its level of cybersecurity. Thus, related work includes integrated cybersecurity risk assessments for the safety of ICS. To compare our work, we propose a summary, presented in the Figure 6, highlighting the key differences and advantages of our method compared to existing methods which are detailed in a second step.

5.1.1. Method differences and advantages

To the best of our knowledge, there is no consensus on the metrics to be used to compare safety-security risk assessments. In this work, we chose to compare our work with the state-of-the-art in terms of comprehensiveness and scalability (the modeled system and its corresponding modeling size). These two criteria allow to assess important features from risk assessment methodologies since a lot of existing methodologies will fall short either not cover all parts required in an evaluation process as described in Section 2 (hence the completeness criterion), or will not be able to properly handle larger system and will only work on toy examples (hence the scalability criterion).We consider that these two metrics are the more relevant for the evaluation of risk assessment methodologies in large industrial control systems. Nevertheless others metrics can be defined like complexity and granularity of the attacker model, ability to generate multi-step attack scenarios, granularity of the model (some methodologies will consider only networks flows, other will go in detail down to program line analysis), specificity (some methodologies are built for a specific use case, other are general). Even if our methodology will be well evaluated with some of these other metrics (for instance, we provide a general methodology with a fine granularity and detailed attacker model) we focus on the risk assessment automation, therefore we consider that the most significative metrics, from this point of view, are scalability and comprehensiveness.

We decided to assess the methods' completeness in line with the ISO 31000 standard risk assessment process. ISO 31000 defines the risk assessment process as a three-stage process to identify, analyze, and evaluate the risk. According to this definition, we only

Method	**System size**	**Modeling size**	**Comprehensiveness**
Winther et al. [23]	Train Leader Telephone System (system including 7 components)	Not provided	Id
Cárdenas et al. [24]	3 sensors and 3 actuators	Not provided (Fortran and MATLAB codes)	Id, An, Ev
Song et al. [25]	PLC-based reactor protection system	No system model	Id
Young and Leveson [26]	Main steam isolation valve of Pressurized Water Reactor	5 vertices and 6 edges	Id
Rocchetto and Tippenhauer [27]	18 sensors and 27 actuators	1000 lines of code (ASLAN++)	Id
Mesli-kesraoui et al. [28]	Valve	107 states (timed automaton)	No cybersecurity assessment
Zhu et al. [29]	8 actuators and 3 sensors	30 vertices and 32 edges (Extended multilevel flow model)	Id, An, Ev
Puys et al. [30]	2 sensors and 2 actuators	Not provided (timed automata)	Id
Cheh et al. [31]	Diamond junction at an end train station	10, 11, 3, 2 states (4 timed automata)	Id
Khaled et al. [32]	5 sensors and 3 actuators	99,220 states (Markov decision process)	Id
Eckhart et al. [33]	Robot cell of a spot welding process	Not provided	Id
Bhosale et al. [34]	3 actuators and 4 sensors	74 nodes and 77 edges (BBN)	Id
Son et al. [35]	PLC-based reactor protection system	No system model	Id, An, Ev
This paper	26 sensors and 23 actuators	52, 30, 36 and 12 states (4 finite state automata)	Id, An, Ev

Id = Identify, An = Analyze, Ev = Evaluate

Table 6: Comparison Summary

found three comprehensive integrated safety-security risk assessments in the state-of-the-art [24, 29, 35].

Table 6 can be split into two groups according to the system size. The first group [23, 25, 26, 28, 29, 30, 32, 34] apply their methodology to small/medium use cases to demonstrate the fine granularity of their safety-security modeling. However, when the modeling size is provided, we observe that the methods do not scale up to large use cases. On the other hand, only one existing work [27] apply their method to large systems that shows the scalability of their modeling. However, this method require extensive time and resources to create the model. Our method offers a balance between the modeling granularity to scale for large systems and the modeling cost in terms of the time and resources required to achieve the risk assessment.

5.1.2. Existing methods overview

In this section, we present each related work included in the Table 6 according to the stages of the risk assessment process they address.

Methods without risk assessment

Mesli-kesraoui et al. [28] (2016) used timed automaton to model the control-command chain, i.e., control programs, supervision interfaces, physical equipment, and human tasks. They then verified use of the UPPAAL model checker usage and safety properties on the model. They applied the methodology on a simple use case of a valve opening task.

Risk identification methods

Winther et al. [23] (2001) adapted the HAZOP method for security purposes according to the following schema: **Pre-Guideword Attribute** *of* **component** *due to* **Post-Guideword** such as Deliberate manipulation of the firewall due to insider. Then, from each expression generated from the schema, they identified the threat, the causes, and the consequences. Finally, the method was applied to a Train Leader Telephone System (supervision of the train traffic) use case.

Song et al. [25] (2012) presented a generic risk assessment method based on Nuclear Power Plant (NPP) standards. The method defines a 6-step workflow. First, they identified the system and modeled the cybersecurity. Then, the system's assets were classified into security levels depending on their criticality to the NPP's safety. Each security level requires specific cybersecurity constraints. Third, a threat analysis was conducted to determine potential attack scenarios against the NPP network architecture, followed by a vulnerability analysis (collecting the relevant CVE). Then security controls were designed to eliminate or mitigate the vulnerabilities collected in the vulnerability analysis. Finally, the cybersecurity design was validated in a pentesting (cybersecurity penetration test) phase.

Young and Leveson [26] (2013) applied the System-Theoretic Process Analysis (STPA) for security purposes, STPA-sec. The first step involves identifying the top events, i.e., vulnerabilities and their corresponding loss event, such as the Release of radioactive materials (vulnerability) which may lead to serious human injury or loss of life and environmental contamination (loss event). Then, a graphical specification of the system's functional controls, called High Level Control Structure (HLCS), is built in order to highlight the lack of constraints (security). The third step aims to identify Unsafe/Unsecure control actions that can lead to the system vulnerabilities identified in the first step

(Release of radioactive materials, for instance). Unsafe/Unsecure control actions are identified according to four behaviors.

- Not providing the control action leads to a hazard.
- Providing the control action leads to a hazard.
- Providing a potentially safe control action, but too early, too late, or in the wrong order
- The control action lasts too long or is stopped too soon.

Afterwards, high level security requirements and constraints are defined from the unsecure/unsafe control actions, such as *Close the main steam isolation valve command must never be provided when there is no rupture or leak*. Finally, causal scenarios are built to violate the safety and security requirements and constraints.

Rocchetto and Tippenhauer [27] (2017) presented a methodology that extends the well-known Dolev-Yao attacker model to consider physical interactions with the process. They then modeled the global state of the system through LTL properties written in ASLAN++ language and checked by the CL-Atse [36] (2006) tool. Finally, they evaluated their methodology on a water treatment plant called SWaT, containing 18 sensors and 27 actuators that required a thousand lines of code to be modeled.

Puys et al. [30] (2018) presented a methodology that utilizes timed automata to model both the attacker profile and the system. The UPPAAL model checker then verifies the system's safety properties and the attacker profile's timed automata composition. Khaled et al. [32] (2020) proposed a way to improve this method by composing each component of the system, i.e., physical entities, software, and people that are defined by their capabilities and behaviors. An analyzer checks the graph model of the interaction and communication between the components to generate attacks based on security properties. The method creates an automaton with 100,000 states when applied to a physical process, including 3 actuators and 5 sensors. Cheh et al. [31] (2018) suggested improving [30] the methodology by incorporating the physical layer interaction into the attacker model.

Eckhart et al. [33] presented an automated cybersecurity risk identification using the AutomationML (AML) exchange format. The authors used AML to gather engineering artefacts that they extended with a cybersecurity library (AMLsec). They then transformed the AML to OWL (Web Ontology Language) so as to create a knowledge base that included the engineering artefacts (transformed to OWL from AML), an ICS security ontology, and security link data (CWE, CVE, etc.). From this knowledge base, they were able to automate (1) the identification of threats and vulnerabilities, (2) attack consequences, and (3) cyber-physical attack graph generation. Finally, they applied their method to the official AML example of a real-world robot cell of a spot welding process.

Bhosale et al. [34] (2023) introduced a graphical model that shows the cyberecurity risk for human and functional safety. They used a Bayesian Belief Network (BBN), i.e., a Direct Acyclic Graph (DAG) with conditional probabilities, to compute risk propagation probability. The BBN was composed of nodes to model the system functions along with edges to model the interdependency between functional safety, human safety, and security issues into the system. The model allowed them to compute cybersecurity risk propagation according to conditional properties set out for each causal relationship (directed edge). The authors applied their method to the Distributing Pro station 5 that performs holding, sorting, and feeding workpiece functions (3 actuators and 4 sensors). The modeling required a BBN, including 74 nodes and 77 edges.

Comprehensive methods

Cárdenas et al. [24] (2011) proposed a risk assessment method based on a process simulation. They simulated a simplified Tennesse-Eastman plant where the process model was implemented in FORTRAN and the control law of the proportional integral controllers was implemented in MATLAB. The authors then simulated integrity and denial of service attacks on the system's sensor network to identify the most effective attack to over-pressure a tank (unsafe state that may lead to an explosion). The simulated system included 3 sensors, 3 actuators, and 4 proportional integer controllers.

Zhu et al. [29] (2018) introduced a dynamic risk assessment model, including a system modeling a multilevel flow model (MFM) that describes the structure and purpose of the industrial system in a graphical way. A multilayer Bayesian network was built from this model to model attack propagation, while a process loss calculation evaluated the consequences in terms of money. Lastly, the risk was computed from the multiplication of probability and loss. The method was applied to a chemical process that included 3 processes: the reaction process (4 valves and 1 pressure sensor), the separation process (2 valves and 1 liquid level sensor), and the extraction process (2 valves and 1 liquid level sensor).

Son et al. [35] (2023) presented a comprehensive cybersecurity risk assessment for the safety of Nuclear ICS. The method used the well-known system theoretic process analysis (STPA) to identify threat scenarios that could violate safety constraints. First, the authors defined accidents and risks to the system (system losses, system hazards, and safety constraints). Then they modeled a threat network for each critical digital asset (CDA) according to a list of threat events (like STRIDE) and attack vectors (from the Nuclear Energy Institute standard NEI 10-09). Third, they modeled Unsafe Control Actions (UCA) by defining different control behaviors and their consequences (cf. Young and Leveson [26], STPA-sec). Following this, the authors analyzed the threat network events that can lead to a UCA from four cyberthreats: in the controller, in the control input, in the control path, and in the control process. Finally, they ranked the cybersecurity risk.

5.2. Automation of IEC 62443-3-2 Standard Risk Assessment

To our knowledge, there is no automated methodology for IEC 62443-3-2 risk assessment. However, Ehrlich et al. [37] (2023) presented a prototypical implementation of a risk assessment process based on six IEC 62443-3-2 requirements (ZCR) through a four-step methodology: Network Segmentation (ZCR 2 and ZCR 3.1), Requirements & Guarantees (ZCR 5.6), Risk Assessment (ZCR 5.1 and 5.2), and Attestation (ZCR 7.1). In [38] (2023), a method for determining the target security level (ZCR 5.6) in accordance with the IEC 62443 standard based on the MITRE ATT&CK framework [10] and the Intel Threat Agent Library [39] is presented. We did not find any information on automating the other requirements.

Fockel et al. [40] (2019) presented a four-step methodology for performing an IEC 62443-compliant threat analysis. Like us, the authors used the Microsoft Threat Modeling Tool to identify the threats to the system. However, they provided no information

[10] https://attack.mitre.org/

about their model apart from "*For compliance with IEC 62443, we tailored the template to the industrial automation domain to account for the domain-specific taxonomy, technologies, and threats*". In addition, our work proposes automating the threat analysis and assessing the cybersecurity risk to safety, unlike [40], which only assesses the cybersecurity risk.

6. Conclusion

In this article, we presented our safety-security risk assessment to help practitioners converge safety and security in ICS. The proposed method is based on a generic risk assessment process (ISO 31000) and can be used to automate the recent and widely used IEC 62443-3-2 standard.

Working from previous studies [5, 6], we devised a method to build a DFD of the system automatically from the PLCOpen file format and a process to map the vulnerabilities identified by the threat modeling tool with the method we designed to identify cybersecurity for system safety. We also presented a risk matrix building method according to the level of risk of the attack scenarios generated. In addition, we showed that through evaluation, our methodology is comprehensive and scalable to large systems. To scale, we developed a threat model less complex than those found in the literature, such as the Dolev-Yao model. Our model allows for generating one-step attack scenarios having a direct impact on local properties of the system, where state-of-the-art methods can include multi-steps attack scenarios on system global properties. Nevertheless, our model can be extended with attack trees in order to integrate multi-steps attacks and enrich attacker's behavior in return for a reduction of the method automation level and its repeatability because attack trees require an extensive time to be achieved and highly rely on expert elicitations.

In a future work, we plan to study the automation of IEC 62443-3-2 steps that are specific to this standard, such as the system partitioning into zones and conduits (ZCR 3) or the determination of the target security level (SL-T) of the ZCR 5.6 in order to completely automate the IEC 62443-3-2 standard process. Moreover, we want to extend our method to include the automation of risk treatment.

7. Acknowledgments

This work was supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (IRT Nanoelec, ANR-10-AIRT-05).

References

[1] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, A. Hahn, S. Saravia, A. Sherule, M. Thompson, Guide to Operational Technology (OT) Security, Tech. Rep. NIST Special Publication (SP) 800-82, Rev.3, National Institute of Standards and Technology, Gaithersburg, MD (2023). doi:10.6028/nist.sp.800-82r3.

[2] IEC 62443-1-1, Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models, International standard, International Electrotechnical Commission, Geneva, CH (Jul. 2009).

[3] IEC 62443-3-2, Security for industrial automation and control systems – Part 3-2: Security risk assessment for system design, International standard, International Electrotechnical Commission, Geneva, CH (Jun. 2020).

[4] ISO 31000, Risk management - Guidelines, International standard, International Organization for Standardization, Geneva, CH (Jun. 2018).

[5] M. Da Silva, M. Puys, P.-H. Thevenon, S. Mocanu, N. Nkawa, Automated ICS template for STRIDE Microsoft Threat Modeling Tool, Proceedings of the 18th International Conference on Availability, Reliability and Security (2023). doi:10.1145/3600160.3605068.

[6] M. Da Silva, M. Puys, P.-H. Thevenon, S. Mocanu, PLC Logic-Based Cybersecurity Risks Identification for ICS, Proceedings of the 18th International Conference on Availability, Reliability and Security (2023). doi:10.1145/3600160.3605067.

[7] IEC 61131-10, Programmable controllers - Part 10: PLC open XML exchange format, International standard, International Electrotechnical Commission, Geneva, CH (Jul. 2019).

[8] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, Y. Halgand, A survey of approaches combining safety and security for industrial control systems, Reliability Engineering & System Safety (2015). doi:10.1016/j.ress.2015.02.008.

[9] IEC 62443-4-1, Security for industrial automation and control systems - Part 4-1 : secure product development lifecycle requirements, International standard, International Electrotechnical Commission, Geneva, CH (Mar. 2018).

[10] IEC 62443-4-2, Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components, International standard, International Electrotechnical Commission, Geneva, CH (Apr. 2019).

[11] IEC 62443-3-3, Industrial communication networks – Network and system security – Part 3-3: System security requirements and security levels, International standard, International Electrotechnical Commission, Geneva, CH (Aug. 2013).

[12] IEC 61131-3, Programmable controllers - Part 3: Programming languages, International standard, International Electrotechnical Commission, Geneva, CH (May 2013).

[13] IEC 61131-5, Programmable controllers - Part 5: Communications, International standard, International Electrotechnical Commission, Geneva, CH (Feb. 2000).

[14] L. Kohnfelder, G. Praerit, The threats to our products, [Online; accessed 24-January-2024] (1999).
URL https://shostack.org/files/microsoft/The-Threats-To-Our-Products.docx

[15] A. Shostack, Threat Modeling: Designing for Security, 1st Edition, Wiley Publishing, 2014.

[16] Microsoft, Microsoft Threat Modeling Tool, [Online; accessed 08-January-2024] (2022).
URL https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool

[17] First, Common vulnerability scoring system version 3.1 : Specification document revision 1, [Online; accessed 14-December-2023] (2019).
URL https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf

[18] First, Common vulnerability scoring system version 4.0 : Specification document, [Online; accessed 14-December-2023] (2023).
URL https://www.first.org/cvss/v4.0/specification-document

[19] J. Provost, J.-M. Roussel, J.-M. Faure, Translating Grafcet specifications into Mealy machines for conformance test purposes, Control Engineering Practice (Sep. 2011). doi:10.1016/j.conengprac.2010.10.001.

[20] Ocanis, open source version of the tool Teloco (Grafcet_to_Automaton), [Online; accessed 22-April-2024] (2021).
URL https://github.com/ocanis/Grafcet_to_Automaton

[21] J. J. Downs, E. F. Vogel, A plant-wide industrial process control problem, Computers & chemical engineering 17 (3) (1993) 245–255.

[22] L. Piètre-Cambacédès, M. Bouissou, Modeling safety and security interdependencies with bdmp (boolean logic driven markov processes), in: 2010 IEEE International Conference on Systems, Man and Cybernetics, 2010, pp. 2852–2861. doi:10.1109/ICSMC.2010.5641922.

[23] R. Winther, O.-A. Johnsen, B. A. Gran, Security assessments of safety critical systems using hazops, in: U. Voges (Ed.), Computer Safety, Reliability and Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 14–24.

[24] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, S. Sastry, Attacks against process control systems: risk assessment, detection, and response, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 355–366.
URL https://doi.org/10.1145/1966913.1966959

[25] J.-G. Song, J.-W. Lee, C.-K. Lee, K.-C. Kwon, D.-Y. Lee, A CYBER SECURITY RISK ASSESSMENT FOR THE DESIGN OF I&C SYSTEMS IN NUCLEAR POWER PLANTS, NUCLEAR ENGINEERING AND TECHNOLOGY (2012) 10.

[26] W. Young, N. Leveson, Systems thinking for safety and security, in: Proceedings of the 29th Annual Computer Security Applications Conference, ACSAC '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 1–8. doi:10.1145/2523649.2530277.
URL https://doi.org/10.1145/2523649.2530277

[27] M. Rocchetto, N. O. Tippenhauer, Towards formal security analysis of Industrial Control Systems, Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (2017). doi:10.1145/3052973.3053024.

[28] S. Mesli-Kesraoui, A. Toguyeni, A. Bignon, F. Oquendo, D. Kesraoui, P. Berruet, Formal and joint verification of control programs and supervision interfaces for socio-technical systems components, (International Federation of Automatic Control (2016). doi:10.1016/j.ifacol.2016.10.603.

[29] Q. Zhu, Y. Qin, C. Zhou, W. Gao, Extended multilevel flow model-based dynamic risk assessment for cybersecurity protection in industrial production systems, International Journal of Distributed Sensor Networks 14 (6) (2018) 155014771877956. doi:10.1177/1550147718779564.
URL http://journals.sagepub.com/doi/10.1177/1550147718779564

[30] M. Puys, M.-L. Potet, A. Khaled, Generation of applicative attacks scenarios against Industrial Systems, Foundations and Practice of Security (2018). doi:10.1007/978-3-319-75650-9_9.

[31] C. Cheh, A. Fawaz, M. A. Noureddine, B. Chen, W. G. Temple, W. H. Sanders, Determining tolerable attack surfaces that preserves safety of cyber-physical systems, 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (2018). doi:10.1109/prdc.2018.00023.

[32] A. Khaled, S. Ouchani, Z. Tari, K. Drira, Assessing the severity of smart attacks in industrial cyber-physical systems, ACM Transactions on Cyber-Physical Systems (2020). doi:10.1145/3422369.

[33] M. Eckhart, A. Ekelhart, E. Weippl, Automated Security Risk Identification Using AutomationML-Based Engineering Data, IEEE Transactions on Dependable and Secure Computing 19 (3) (2022) 1655–1672. doi:10.1109/TDSC.2020.3033150.
URL https://ieeexplore.ieee.org/document/9237126/

[34] P. Bhosale, W. Kastner, T. Sauter, Integrated safety-security risk assessment for production systems: A use case using bayesian belief networks, in: 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), 2023, pp. 1–6. doi:10.1109/INDIN51400.2023.10217926.

[35] K.-S. Son, J.-G. Song, J.-W. Lee, Development of the framework for quantitative cyber risk assessment in nuclear facilities, Nuclear Engineering and Technology 55 (6) (2023) 2034–2046. doi:https://doi.org/10.1016/j.net.2023.03.023.
URL https://www.sciencedirect.com/science/article/pii/S1738573323001328

[36] M. Turuani, The CL-ATSE protocol analyser, Lecture Notes in Computer Science (2006). doi:10.1007/11805618_21.

[37] M. Ehrlich, A. Bröring, H. Trsek, J. Jasperneite, C. Diedrich, Evaluation concept for prototypical implementation towards Automated Security Risk Assessments, IEEE 28th International Conference on Emerging Technologies and Factory Automation (2023). doi:10.1109/etfa54631.2023.10275455.

[38] M. Ehrlich, A. Bröring, C. Diedrich, J. Jasperneite, W. Kastner, H. Trsek, Determining the target security level for Automated Security Risk Assessments, 2023 IEEE 21st International Conference on Industrial Informatics (2023). doi:10.1109/indin51400.2023.10217902.

[39] T. Casey, Threat agent library helps identify information security risks, Intel White Paper 2 (2007).

[40] M. Fockel, S. Merschjohann, M. Fazal-Baqaie, T. Förder, S. Hausmann, B. Waldeck, Designing and integrating IEC 62443 compliant threat analysis, Communications in Computer and Information Science (2019). doi:10.1007/978-3-030-28005-5_5.