Automated Security Risk Identification Using AutomationML-based Engineering Data

Matthias Eckhart, Andreas Ekelhart, and Edgar Weippl, *Member, IEEE*

Abstract—Systems integrators and vendors of industrial components need to establish a security-by-design approach, which includes the assessment and subsequent treatment of security risks. However, conducting security risk assessments along the engineering process is a costly and labor-intensive endeavor due to the complexity of the system(s) under consideration and the lack of automated methods. This, in turn, hampers the ability of security analysts to assess risks pertaining to cyber-physical systems (CPSs) in an efficient manner. In this work, we propose a method that automatically identifies security risks based on the CPS's data representation, which exists within engineering artifacts. To lay the foundation for our method, we present security-focused semantics for the engineering data exchange format AutomationML (AML). These semantics enable the reuse of security-relevant know-how in AML artifacts by means of a formal knowledge representation, modeled with a security-enriched ontology. Our method is capable of automating the identification of security risk sources and potential consequences in order to construct cyber-physical attack graphs that capture the paths adversaries may take. We demonstrate the benefits of the proposed method through a case study and an open-source prototypical implementation. Finally, we prove that our solution is scalable by conducting a rigorous performance evaluation.

Index Terms—cyber-physical systems, information security, AutomationML, security modeling, security risk assessment, industrial control systems, IEC 62443.

1 Introduction

Cyber-physical attacks give rise to safety concerns, as they can be launched in the cyberspace with consequences in the real world, potentially endangering human health or the environment. The devastating impact of past security incidents involving cyber-physical systems (CPSs), more specifically, industrial control systems (ICSs), raised awareness and continually motivates research in this field. In the light of the ever-evolving threat landscape, asset owners seek to improve the security of the CPSs they are operating, by upgrading their so-called "brownfield" facilities with security controls, while systems integrators, in coordination with vendors of industrial components, are primarily aiming to carry out "greenfield" engineering projects with security in mind. However, establishing security as a 'first-class citizen' in the engineering process has been identified by scholars as a primary challenge [1]. Overcoming this challenge requires the introduction of dedicated security activities, which need to be conducted along the engineering process [2]. Security risk management efforts are vital for adopting such a security-enhanced engineering process, as security risks need to be addressed in a cost-effective manner. Yet, the lack of adequate support for performing risk assessment tasks represents a fundamental barrier to adopting such a security-enhanced engineering process.

First and foremost, engineering data formats and modeling languages need to provide the means to represent security information, since they can be considered as the backbone of digitized engineering workflows and may therefore serve as a primary source for assessing security risks pertaining to the CPSs being engineered. One such data format that currently lacks the semantic modeling concepts to express security know-how in a systematic and structured manner is AutomationML (AML). AML is a standardized XML-based format used for the exchange of engineering data, including information related to the topology, geometry, kinematics, and behavior of industrial components [3]. This markup language was designed with the objective to improve the data exchange between engineering tools, but its field of application evolved to the modeling of CPSs [4]. Seamlessly integrating security-related information into AML artifacts may not only reduce barriers for engineers to take security properties of CPSs into consideration, but also enables additional security-enhancing use cases, such as automating security risk assessments.

Second, in light of the gradual adoption of the IEC 62443 series of standards for industrial security, systems integrators are in need of an automated risk identification method that supports them in carrying out security risk assessments according to IEC 62443-3-2 [5]. This method must be tightly integrated into the engineering process to utilize existing engineering knowledge to the full extent [6]; thereby, establishing security as an integral part of CPS engineering workflows. It should further identify relationships among the detected security risks in the form of cyber-physical attack chains or graphs [7] and thereby provide the foundation for a subsequent quantitative risk analysis [6].

This work aims to address these gaps by presenting a method for the automated identification of security risks in line with the IEC 62443-3-2 [5]. This method is based on an ontological security modeling approach and leverages the

M. Eckhart, A. Ekelhart, and E. Weippl are with SBA Research, 1040 Vienna, Austria and the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle, TU Wien, 1040 Vienna, Austria. E-mail: {MEckhart, AEkelhart, EWeippl}@sba-research.org.

introduced security-focused semantic constructs for AML engineering data. Our novel approach automatically identifies vulnerabilities in the engineered systems, the corresponding security threats, and the potential consequences of attacks, allowing to build comprehensive cyber-physical attack graphs (CPAGs) that are vital to understanding security risks pertaining to CPSs. Given its automated nature and the fact that engineering knowledge is obtained from already existing artifacts, this approach significantly improves the efficiency and effectiveness of the risk identification step within the process specified in IEC 62443-3-2 [5]. In this way, risk analysts can analyze, evaluate, and initiate the treatment of the identified security risks already at the very beginning of the CPS lifecycle, thereby supporting a security-by-design engineering process.

The novelty and main contributions of the paper at hand can be summarized as follows:

- We present a method for the automated identification of security risks based on engineering data, which follows the IEC 62443-3-2 [5].
- We propose a security modeling concept for AML named *AMLsec* that engineers and engineering tool vendors can adopt for the adequate representation of the CPSs' security properties.
- We introduce the notion of cyber-physical attack graphs (CPAGs), i.e., a variant of host-based attack graphs (AGs) that systems integrators can apply to gain insights into possible multistage cyber attacks launched against components of the plant topology with the objective of causing physical damage.
- We show the feasibility of our developed concepts by providing an open-source prototypical implementation, demonstrate the benefits of our contribution in a case study, and finally evaluate the performance and scalability of our solution.

The remainder of this paper is divided into four sections. Section 2 gives a synopsis of relevant literature. In Section 3, we present the security modeling concept for AML (AMLsec), discuss how sources of security risks and consequences can be automatically obtained, and introduce the concept of CPAGs. Next, in Section 4, we describe a case study that is used to test the practicality of the introduced method. Subsequently, in Section 5, we present the results of our performance evaluation. Finally, in Section 6, we summarize our findings and suggest ideas for further research.

2 Background & Related Work

In this section, we provide background information for our method and review relevant literature in order to place our work in context. First, we give an overview of AML, as we utilize this data exchange format to obtain engineering information. Next, we provide an introduction to risk assessment in the context of the IEC 62443 series. After that, we discuss related approaches for conducting security analyses, which are also based on AML. Then, we analyze existing security modeling languages and their application in the context of CPS engineering. Subsequently, we describe existing research on attack graph generation and further motivate the need for CPAGs. Finally, we briefly review security ontologies and explain their relevance in automating the assessment of CPSs security risks.

2.1 AutomationML

AutomationML (AML) is a data exchange format that aims to remedy the issues of heterogeneous engineering networks, as it has been designed with the objective to achieve interoperability within engineering toolchains [3], [8]. This XML-based data format is developed and promoted by the AutomationML association[1] and has been standardized in the IEC 62714 series. AML is based on the Computer Aided Engineering Exchange (CAEX) data format, according to the IEC 62424 [9], and therefore adopts the object-oriented paradigm [8]. More specifically, AML provides the means to represent engineering information in four different ways, viz., (i) as part of a hierarchy by means of CAEX to describe the plant/system topology, components, and network-related information, (ii) as geometry and kinematic models by means of COLLADA, (iii) as control logic by means of PLCopen XML, and (iv) as CAEX-based references to describe relationships among objects [8]. The unified representation of know-how concerning the CPSs to be engineered is what makes AML particularly attractive for obtaining security-relevant information from respective artifacts. Moreover, prior research indicates that AML does not only address data exchange issues, but also supports a model-based engineering approach [4]. As we will show in the work at hand, the AML modeling concepts can be augmented with security-related constructs to provide a common data basis that enables the automated identification of security risks and thereby fosters security-aware CPS engineering.

2.2 Security Risk Assessment as per IEC 62443-3-2

Part 3-2 [5] of the IEC 62443 series specifies a process that organizations can adopt for assessing security risks pertaining to ICSs. This process is composed of a set of Zone and Conduit Requirements (ZCR) that need to be implemented to establish the zones and conduits model and assess security risks. In a nutshell, the respective workflow consists of the following tasks [5]: (ZCR-1) Identification of the System under Consideration (SuC), (ZCR-2) Initial security risk assessment, (ZCR-3) Partitioning of the SuC into zones and conduits, (ZCR-4) Assessment whether high--level risks exceed tolerable risks, (ZCR-5) Detailed security risk assessment, and (ZCR-6) Documentation of security requirements, assumptions, and constraints. In general, risk identification is a subprocess of risk assessment and focuses on finding sources of risks and their potential consequences, providing the basis for subsequent risk analysis and risk evaluation [10]. Thus, our method is applicable to multiple steps of the workflow defined in IEC 62443-3-2 [5], particularly ZCR-2, ZCR-3, and ZCR-5.

2.3 AutomationML-based Security Analysis

The notion of utilizing engineering data for security-enhancing purposes has been explored in several works,

1. https://www.automationml.org

such as [11], [12], [13] which discuss how a knowledge-based system in conjunction with AML artifacts can be used for automatically identifying security issues in the plant design. The approach described in these papers connects (i) plant-specific information as specified in AML artifacts, and (ii) engineering and information security domain knowledge modeled with the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL). As explained in [11], [12], [13], the CAEX structure is first extracted from the AML file to obtain information about the plant topology comprising the physical network and devices. Then, the plant topology model is augmented with engineering domain knowledge to add, for example, vendor-specific information that is not available in engineering tools [12]. Furthermore, the know-how used for the security analysis is modeled with OWL/SWRL and can be obtained from various sources, such as standards and guidelines, vulnerability databases, and CERT advisories [14]. To apply the security domain knowledge, the authors developed specific operators for addressing CAEX elements of the plant topology model using SWRL-based rules. It is worth noting that the plant topology information is not converted to an ontological representation; instead, the designed operators are used by the custom inference engine to fetch the CAEX model existing in the AML files and interpret its elements in conjunction with the ontology [11], [13]. As discussed in [11], [13], the authors abstained from a CAEX-model-to-ontology conversion, as they deemed the (bidirectional) transformation process to be complex and potentially error-prone. Thus, they implemented the proposed concept based on the AutomationML Engine to directly access the CAEX model and programmatically evaluate the defined rules to perform the security assessment.

In addition to the efforts made by researchers described above, the German-based security consultancy admeritia has also made attempts to establish security-by-design engineering by means of AML. Fluchs and Rudolph presented a concept named *Layered Blueprints* [15], [16] that can serve as a "thought model" for engineers to consider security aspects during engineering activities. To embed the thought model into the engineering process and make it fit for use, a data model based on AML has been developed [17]. This model focuses on the network characteristics and respective use cases, such as the programming of a programmable logic controller (PLC), to capture the semantics relevant for security engineering [17].

Overall, some progress in AML-enabled security analysis has already been made. Nevertheless, the two reviewed approaches leave ample room for further improvement. While the security analysis approach discussed in [11], [12], [13] seems to primarily focus on the identification of threats using a rule-based (if—then) logic, admeritia's thought/data model, which appears to be still in development, utilizes AML as a common interface for OT engineers and security analysts. The findings of the survey paper [6] highlight that there is a need for methods to automate the identification of security risk sources and potential consequences based on engineering data. We argue that security semantics for AML are indispensable for automating the identification of security risk sources and potential consequences based on AML artifacts. Although our method requires the annotation of security-relevant information by engineers or security analysts in AML artifacts, the provided capabilities support a security risk assessment as per the IEC 62443-3-2 [5], yield CPAGs, and even lay the foundation for a quantitative consideration of risks (e.g., by means of AG-based security metrics). Furthermore, in contrast to the approach discussed in [11], [12], [13], our proposed method transforms the entire engineering data representation present in AML to OWL, allowing the use of established semantic technologies, including existing semantic reasoners to infer knowledge.

2.4 Languages for Security-Aware Engineering

In the past years, several language proposals for security-aware engineering have been presented, suggesting an increased interest in utilizing models and engineering data representations for the purpose of improving the security of CPSs. In the following, we summarize the key differences among relevant works.

UMLsec is a security extension for the Unified Modeling Language (UML) that was realized by means of a UML profile [18]. It is applicable to various UML diagram types, making it versatile in multiple applications [19]. To support users in performing security analyses, tools have been developed that can automatically inspect UMLsec models and formally verify the therein specified security properties [19]. *SecureUML* also uses a security-enriched UML profile, albeit it is limited to class diagrams [20]. Its initial use case focused on the generation of access control policies based on the role-based access control (RBAC) model [20], yet it has been extended in a subsequent work [21] to adapt it to security risk modeling. Robles-Ramirez et al. [22] introduced *IoTsec*, i.e., a UML security modeling extension intended to be used along the engineering lifecycle of Internet of Things (IoT) devices. IoTsec is composed of a nomenclature, consisting of fifteen security requirements, and UML diagrams that have been extended for modeling IoT-specific security know-how. In addition to UML, researchers have also proposed language extensions for SysML. For instance, Apvrille and Roudier [23] present *SysML-Sec*. The introduced stereotypes (e.g., security requirement) and diagrams (e.g., attack tree) enable users to directly integrate security-relevant information into the model-based systems engineering workflow and formally verify annotated security properties of systems [23]. Further, the SysML extension proposed by Oates et al. [24] is specifically tailored to designing secure ICSs. Their language extension comprises two profiles: one can be used to model threats and how they could pose a security risk to assets; the other profile can be used to model the interactions between assets and data. Contrary to SysML-Sec, the two profiles developed by the authors of [24] focus more on the architectural aspects of ICSs, rather than the system design of the therein employed industrial components. The SysML extension introduced in [25] shifts the focus of analyzing ICS security issues away from graphical modeling to formal reasoning. The authors have utilized SysML models of ICSs as a knowledge source for feeding the inductive definition programming (IDP) framework [26], which then can be used for automatically identifying vulnerabilities. In [27], Lemaire et al. extend their initial work on using formal reasoning for improving ICS security by presenting a prototype and evaluating their approach by means of a real-world case study featuring an industrial hatchery.

It is also worth noting that researchers designed languages that are not based on any pre-existing modeling language, such as UML or SysML. One such standalone language is the *Cyber Security Modeling Language (CySeMoL)* [28] (later extended in [29]) that adopts a probabilistic relational model (PRM) [30] for security risk analysis [31], allowing to represent the inputted system architecture as an object model and generate a Bayesian network from it. *S-cube* [32] is another standalone modeling approach that utilizes the Figaro language [33] and covers both security and safety. S-cube receives as an input a model of the system architecture under consideration and automatically outputs attack and failure scenarios that can then be analyzed in a qualitative and quantitative manner.

This overview on related work confirms that security-aware engineering, in connection with risk assessment, is an active field of research. Most of the reviewed works do not provide standalone security modeling languages, but rather extend UML or SysML, and are therefore diagrammatic. It is evident that by extending well-established modeling languages, the already developed models can be directly reused for security-enhancing purposes, such as performing risk assessments. Thus, creating a domain-specific security modeling language that requires users to (re)input the system model may represent a barrier to adopting a security-aware engineering approach. However, the reviewed standalone languages [28], [29], [32] appear to be more geared toward security risk analysis, with a strong focus on estimating the level of risk in quantitative terms.

Based on the presented review of the state of the art, we highlight the following differences between the above-mentioned language proposals and our method:

- Current language extensions place a special emphasis on designing secure (sub)systems, rather than securely integrating them into an industrial automation solution. While vendors of industrial components can adopt these extensions for their engineering, systems integrators are currently left with no alternative. This underscores the need for security modeling extensions for tools and data formats to be found in typical engineering toolchains of systems integrators. AML constitutes a solid candidate for such an extension, given the fact that it is a standardized engineering data exchange format and also suitable for modeling purposes.
- The reviewed language constructs can be used to formally verify security properties (e.g., SysML-Sec [23]) or conduct risk assessments using the complete plant architecture as the system under consideration (e.g., CySeMoL [28]). However, our method is intended to be used incrementally from basic to detailed engineering for the purpose of identifying sources of risks and potential consequences as early as possible. Particular attention has been paid to the IEC 62443-3-2 [5] risk assessment procedure.
- The seamless integration and automated nature of our method represents a further key characteristic. Although users need to augment the engineering data representation with additional information, this task fits naturally into the workflow, and modeling effort is kept to a minimum owing to the adoption of standard AML concepts and entities. Furthermore, the open-source prototype can be directly integrated into the engineering toolchain without the risk of interfering with other tools.

2.5 Attack Graph Generation

Attack graphs and, particularly, their efficient and automated construction is a well-studied topic in the security modeling area [34]. Although attack graphs have already been thoroughly investigated by the CPS community [35], we argue that an ontology-driven approach for the automated generation of attack graphs based on engineering data, which capture the cross-domain effects of cyber-physical attacks, is currently missing. To better position our work in relation to other attack graph generation approaches, we classify our proposal according to Kaynar's taxonomy [36] and reference relevant literature:

- *Reachability analysis phase:* Kaynar further categorized the reachability of hosts according to the *scope* of the reachability analysis and the *content* used to determine reachability [36]. Our attack graph generation approach is primarily intended to be used during the engineering phase of CPSs; hence, reachability is computed based on the network topology modeled by engineers. In this sense, our approach computes *whole network reachability* based on *trust relationships*, which are modeled as logical or physical connections in AML artifacts. Other approaches that fall into the same categories and were evaluated in the context of CPSs, such as ADVISE [37], [38], do not directly reuse the plant specification that is created during engineering but often require additional effort to supply proper inputs to the AG generation mechanism.
- *AG modeling phase:* The taxonomy presented in [36] includes the classification criteria *attack model* and *attack graph model* to consider the conditions of atomic attacks and the structure of the AG. Our approach generates *host-based AGs* with relaxed preconditions and postconditions that abstract the adversary model. In [39], [40], [41] models with a similar host-based structure were proposed. It is worth noting that we base our AG on the model described by Zhong et al. [39], which also features a vulnerability library and the weighting of edges. However, a distinguishing characteristic of our proposed AG is that it takes the cyber-physical criticality of plant components into account by weighting vertices based on the determined type of impact (e.g., physical damages). This feature allows to prune the AG in a way that retains the most relevant attack paths, which start in the cyber domain and induce physical consequences.
- *AG core building phase:* Kaynar classified AG construction from two different angles, namely based on the *determination method* and the *pruning method* [36]. We utilize an ontological approach to build a full AG and programmatically prune it afterwards. Other researchers have also investigated how a semantic, ontological approach can enable the generation [42] or improve the

analysis of attack graphs [43], [44]. In addition to the differences in the implementation of AG construction (e.g., Wu et al. [42] used SWRL and Jess [45]), our approach is embedded into a full risk identification solution, specifically targeted at the CPS domain.
- *Uses of AGs:* Our main purpose of generating attack graphs is to facilitate and improve the identification of security risks for engineers, who typically do not have a security background.

2.6 Information Security Ontologies

As already indicated, we make use of a formal knowledge model for automating security risk assessments. Prior research in the area of formalizing information security know-how has shown that ontological approaches can facilitate automated security risk assessments [46]. To the best of our knowledge, only the works [47], [48] discussed the use of ontologies in the context of ICSs thus far. Tebbe et al. [47] described the lifecycle of ICS-specific security knowledge and proposed an ICS security ontology. Wolf et al. [48], on the other hand, presented an ontology-based modeling approach for security analysis. None of the works cover ontological security modeling in connection with AML-to-OWL transformation for the purpose of automating security risk identification.

3 Method

Our novel method aims to automate those tasks of the risk assessment workflow, according to the IEC 62443-3-2 [5], that concern the risk identification process. More specifically, our method supports individuals in meeting the ZCR defined in the IEC 62443-3-2 [5] in the following ways: First, engineers select the AML data representation created during basic or detailed engineering that should be used as the SuC. All the assets contained in the user-supplied AML artifact, which may only represent a subset of the modeled industrial plant, will be considered for risk identification; thus, clearly delineating the scope of the assessment (ZCR-1). Second, our method supports the initial cybersecurity risk assessment by automatically identifying assets involved in hazardous and safety-critical processes, which are indicative of attack targets with a high impact potential and are therefore considered as part of worst-case scenarios (ZCR-2). Third, our proposed method automatically performs a validation according to the ZCR-3.2–3.6 in order to detect flaws concerning the partitioning of the SuC into zones and conduits. Fourth, our method yields a list of the identified threats (ZCR-5.1) and vulnerabilities (ZCR-5.2). Fifth, the proposed approach automatically determines the type of potential consequences (ZCR-5.3), viz., (i) hazards (to personnel, to goods, or to the environment), (ii) breach (breach of intellectual property or data breach), (iii) business interruption, (iv) regulatory non-compliance, and (v) financial loss. In addition, CPAGs can be generated from the identified security risks to represent relations among vulnerabilities in plant components that may indicate possible paths of adversaries.

Fig. 1 provides an overview of our proposed method. As part of the engineering process, engineering artifacts are developed and exchanged. Data formats, such as AML, are used to model and exchange the engineering data representation. In step ❶, engineers annotate and model security-relevant information with the AML security extension libraries (AMLsec), which we introduce in this work. Then, in step ❷, the AMLsec-augmented plant know-how is transformed to OWL. The next step, step ❸, is the expansion of the model realized by merging the engineering data with the security ontology [46], our developed ICS security ontology, and linked open security data to build the Knowledge Base (KB) from the initially separated data sources. Furthermore, as part of this step, the engineering data is validated to ensure that the model is fit for executing the risk identification steps. Finally, in step ❹, the risk identification is automatically performed by executing the developed security rules and queries against the KB. After the security risks have been identified, we automatically model attacks in a graph structure with the CPAG ontology, enabling users to quickly spot critical attack paths that may lead to a physical impact.

In the following subsections, we discuss each part of our method in detail.

3.1 Engineering Data Representation

The engineering data representation comprises system-dependent know-how concerning the CPS(s) to be engineered and constitutes the SuC for the automated risk identification. As discussed by Schleipen and Drath [53], engineers work with this representation in three different ways, viz., with a focus on products, processes, and resources (PPR concept). Since this is an established concept in the engineering domain, the proposed security risk identification method also incorporates these three views. Next, we explain how the presented method integrates the engineering data representation.

3.1.1 Engineering Artifact

As already mentioned, we obtain the know-how concerning the engineered CPS directly from AML artifacts that are exchanged among the tools within the engineering chain. Since AML utilizes CAEX [9], the plant topology is modeled in an object-oriented manner by means of the concepts (i) `InstanceHierarchy` (instances of objects), (ii) `SystemUnitClassLib` (libraries of components for reuse), (iii) `RoleClassLib` (libraries of roles that define semantics), (iv) `InterfaceClassLib` (libraries of interfaces for modeling relations between objects), and (v) `AttributeTypeLib` (libraries of objects' attribute types) [3]. Libraries in AML are vital to our automated risk identification method, as they provide an information mapping mechanism that enables the semantic interpretation of plant topology data in OWL. To harness community resources and ensure a seamless integration of our method into the engineering workflow, we utilize standard AML libraries. In particular, the role class and interface class libraries specified in the AML white papers part 1 [54], part 2 [55], and part 5 [56] are employed in our method. However, these libraries lack certain modeling constructs that engineers require to semantically enrich security-relevant elements of the plant topology (e.g., zones and conduits,

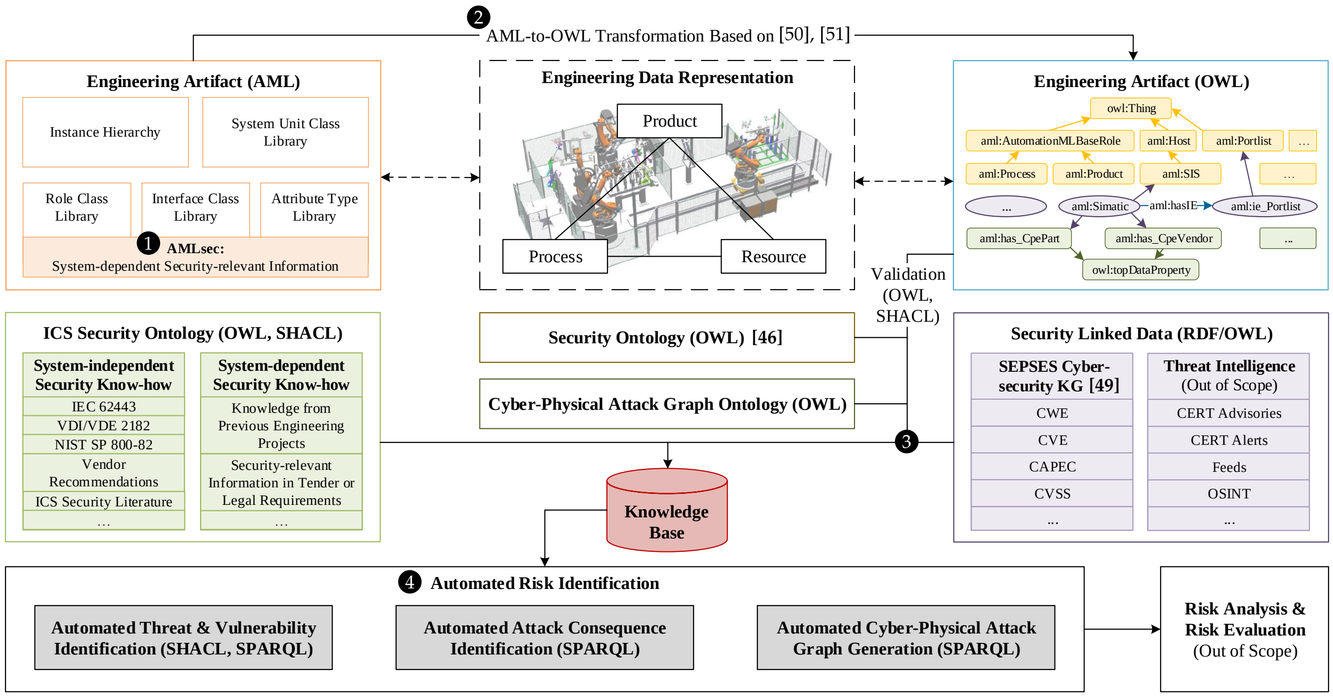

2 AML-to-OWL Transformation Based on [50], [51]
Engineering Artifact (AML)
Instance Hierarchy
System Unit Class Library
Role Class Library
Interface Class Library
Attribute Type Library
1 AMLsec:
System-dependent Security-relevant Information
Engineering Data Representation
Product
Process
Resource
Engineering Artifact (OWL)
owl:Thing
aml:AutomationMLBaseRole
aml:Host
aml:Portlist
…
aml:Process
aml:Product
aml:SIS
…
...
aml:Simatic
aml:hasIE
aml:ie_Portlist
aml:has_CpePart
aml:has_CpeVendor
...
owl:topDataProperty
Validation (OWL, SHACL)
ICS Security Ontology (OWL, SHACL)
System-independent Security Know-how
IEC 62443
VDI/VDE 2182
NIST SP 800-82
Vendor Recommendations
ICS Security Literature
…
System-dependent Security Know-how
Knowledge from Previous Engineering Projects
Security-relevant Information in Tender or Legal Requirements
…
Security Ontology (OWL) [46]
Cyber-Physical Attack Graph Ontology (OWL)
3
Knowledge Base
Security Linked Data (RDF/OWL)
SEPSES Cyber-security KG [49]
CWE
CVE
CAPEC
CVSS
...
Threat Intelligence (Out of Scope)
CERT Advisories
CERT Alerts
Feeds
OSINT
...
4 Automated Risk Identification
Automated Threat & Vulnerability Identification (SHACL, SPARQL)
Automated Attack Consequence Identification (SPARQL)
Automated Cyber-Physical Attack Graph Generation (SPARQL)
Risk Analysis & Risk Evaluation (Out of Scope)

Fig. 1: Overview of our AutomationML-based risk identification method (robot cell illustration taken from [52]).

security devices, security configurations). To eradicate this issue, AMLsec is implemented as a set of libraries comprising role classes, interface classes, and attribute types that engineers (or even the engineering tools by themselves) can use to represent the security-relevant information of plant components.

3.1.2 *AMLsec: AutomationML Security Extension Libraries*

As indicated above, AMLsec is modeled as a (i) `RoleClassLib`, (ii) `InterfaceClassLib`, and (iii) `AttributeTypeLib` to extend the AML base libraries. We provide a set of elements for each of the three AMLsec libraries, but they can be freely extended by users to cover specialized plant components and thereby further expand the risk identification scope.

In AML, role classes associated to CAEX `InternalElements` are used to model the functionality of components in an abstract, implementation-independent manner [54]. Hence, `AMLsecRoleClassLib` is composed of role classes that shall be associated to instances representing architectural segments (e.g., zones), assets (e.g., IT/OT systems), infrastructural components (e.g., wires), network protocols (e.g., OPC UA), or configuration data (e.g., security policies). In line with the rationale behind AML role classes, `AMLsecRoleClassLib` has been designed to be generic and flexible, ensuring that the included role classes are compatible with varying plant topologies and diverse implementations of components.

AML interface classes are used to describe the semantics of relations between instances, which are modeled by means of CAEX `ExternalInterfaces` and `InternalLinks` [54]. Based on the communication system modeling concepts outlined in the AML white paper part 5 [56], we created the `AMLsecInterfaceClassLib`, which includes interface classes that shall be used for describing physical (e.g., Ethernet-based) and logical (protocol-dependent) connections. The physical and logical connections in AML refer to the OSI reference model layers 1–2 and 3–7, respectively [56]. Given the importance of network security aspects to risk identification, the semantic representation of different connection types are therefore covered by this library.

In the second edition of the AML standard, i.e., IEC 62714-1:2018 [57], CAEX 3.0 according to IEC 62424:2016 [9] was introduced, providing the means to define libraries composed of attribute types that can be referenced by instances to model attribute semantics. We created the library `AMLsecAttributeTypeLib` to define attribute types that shall be used to describe security-relevant implementation-specific characteristics. For instance, we created attribute types for the Common Platform Enumeration (CPE) naming scheme to automatically map Common Vulnerabilities and Exposures (CVE) to vulnerabilities in IT/OT systems. Since some engineering tools already provide integrated data portals that allow engineers to download additional information about industrial components (e.g., data sheets, wiring diagrams, documentation) from vendor databases, we envision that, in the future, the CPE attribute values will be automatically populated by the tools themselves upon the AML export. In addition to the CPE attribute types, we included types that shall be applied to configuration data (e.g., know-how protection for control logic blocks).

An excerpt of AMLsec, which contains the definition of security-relevant role classes, interface classes, and attribute types, is shown in Listing 1. These three security extension

```
<RoleClassLib Name="AMLsecRoleClassLib">
 ...
 <RoleClass Name="Zone">
   ...
   <RoleClass Name="OTZone"
   ↪  RefBaseClassPath="AMLsecR.../Zone">
      <RoleClass Name="OperationsSupportZone"
      ↪  RefBaseClassPath="AMLsecR.../Zone/OTZone" />
   ...
   </RoleClass>
 ...
</RoleClassLib>
<InterfaceClassLib Name="AMLsecInterfaceClassLib">
 ...
 <InterfaceClass Name="LogicalEndpoint">
   <InterfaceClass Name="LogicalEndpointOPC-UA"
   ↪  RefBaseClassPath="AMLsecI.../LogicalEndpoint" />
   ...
 </InterfaceClass>
</InterfaceClassLib>
<AttributeTypeLib Name="AMLsecAttributeTypeLib">
 ...
 <AttributeType AttributeDataType="xs:string"
 ↪  Name="CpePart" />
 ...
</AttributeTypeLib>
```

Listing 1: Excerpt of AMLsec.

libraries, in combination with the AML base role class and interface class libraries [54], [55], [56], provide the semantic modeling constructs that need to be applied in engineering activities to make the automated risk identification with our method possible. Plant components that have been modeled as described may also be bundled as part of a `SystemUnitClassLib` for reuse purposes (e.g., a PLC of a certain type including its configuration). It is also worth pointing out that the security-relevant information contained in the AML artifact (and modeled by means of AMLsec) only concerns the system to be engineered and represents therefore merely a fraction of the security know-how in the KB. The rationale behind this is not only to prevent redundant data in engineering artifacts but also to unburden engineers from the responsibility of acquiring specialized security know-how. Furthermore, we want to stress that AMLsec neither replaces other, user-supplied libraries, nor interferes in any manner with existing engineering practices or tools.

3.2 Ontological Security Modeling

The semantic KB comprises the information contained in the engineering artifact, the ICS security ontology, and the security linked data (cf. Fig. 1).

The engineering knowledge that exists in AML is transformed to OWL by applying the conceptual mapping approach proposed by Hua and Hein [50], [51]. In a nutshell, AML `RoleClass`es and `InterfaceClass`es are mapped to OWL classes, AML `SystemUnitClass`es, `InternalElement`s, and `ExternalInterface`s are mapped to OWL individuals, AML relationships (e.g., the hierarchical structure) are mapped to OWL object properties, and AML attributes are mapped to OWL data properties [50], [51]. We extended this approach by mapping AML `InternalLink`s to OWL individuals and mapping their relationships to OWL object properties in order to retain the referenced link partners. Note that the AML-to-OWL transformation process is unidirectional. The implementation of a unidirectional flow was an intentional design decision since we focus on automating the identification of security risks without changing the underlying engineering data.

After transforming the AML engineering artifact to OWL, we define axioms in the terminological box (TBox) $\mathcal{T}$ to establish the semantic mappings between `RoleClass`es contained in an AML base library and the ones contained in the respective AMLsec library. For instance, in Description Logics (DLs), $\mathsf{PLC} \sqsubseteq \mathsf{Host}$ captures the semantic relation between the AML role classes `PLC` contained in `AutomationMLCSRoleClassLib` and `Host` contained in `AMLsecRoleClassLib`. Thus, the introduction of AMLsec replacement roles was not required, meaning that engineers can continue using the role classes included in AML base libraries. Moreover, we define axioms in $\mathcal{T}$ to eradicate the lack of multiple inheritance in `AMLsecRoleClassLib`. To give an example, the AMLsec role class `Firewall` inherits directly from `SecurityDevice` but not from `NetworkDevice`; thus, requiring the axiom definition $\mathsf{Firewall} \sqsubseteq \mathsf{NetworkDevice}$ for the correct semantic representation of this network security device. Further axioms in the role box (RBox) $\mathcal{R}$, such as $\mathsf{hasRefPartner} \equiv \mathsf{hasRefPartnerSideA}^-$ and $\mathsf{hasRefPartner} \equiv \mathsf{hasRefPartnerSideB}^-$, are defined merely for convenience.

Moreover, we perform validation checks to detect modeling errors that would thwart the automated security risk identification. Researchers have already studied an ontology-based validation of engineering data and achieved promising results (cf., for instance, [58]). Thus, we validate the engineering data in two different ways. First, by expanding $\mathcal{T}$ and $\mathcal{R}$ to let reasoners detect semantic inconsistency errors. Second, by using the Shapes Constraint Language (SHACL)[2] to validate the shape that the engineering data should have.

To give two examples: (i) Reasoners can check whether PLC security protection mechanisms were incorrectly added to any instances other than PLC programs based on the definition of the axioms

$$\begin{aligned} \mathsf{PLCProgram} &\sqsubseteq \mathsf{Program} \\ \text{for each } B \in M \setminus \{\mathsf{Program}\}, B \sqcap \mathsf{PLCProgram} &\sqsubseteq \bot \\ \exists \mathsf{hasCopyProtection}.\top &\sqsubseteq \mathsf{PLCProgram} \\ \exists \mathsf{hasKnowHowProtection}.\top &\sqsubseteq \mathsf{PLCProgram}\,, \end{aligned}$$

where M denotes the set of the most general concepts of the engineering ontology. (ii) Incorrectly modeled disjoint union relations of network protocols (e.g., an instance is modeled as a Modbus and OPC protocol) are detected by reasoners by defining the axioms

$$\bigsqcup_{i<j} \mathsf{Protocol}_i \sqsubseteq \mathsf{Protocol}$$
$$\mathsf{Protocol}_i \sqcap \mathsf{Protocol}_j \sqsubseteq \bot \text{ for all } 1 \leq i < j \leq n\,.$$

Additionally, we created SHACL shapes to validate the modeled know-how against rules that fit to a closed-world setting. For example, PLC programs are only applicable to PLCs, conduits must represent a wireless or wired connection, and (physical) port lists must include physical interfaces (e.g., Ethernet sockets).

The ICS security ontology comprises security know-how applicable to the industrial domain. It includes know-how

2. https://www.w3.org/TR/shacl

from system-independent and system-dependent security data sources, which are both provided and maintained by ICS security analysts. In line with the principle of knowledge reuse, and the ICS security knowledge lifecycle proposed by Tebbe et al. [47], we adopted a layered approach that utilizes the security ontology presented in [46] as the middle ontology layer. We adapted this security ontology to consider attack consequences by expanding $\mathcal{T}$ and $\mathcal{R}$:

$$\begin{aligned}
\text{for each } D \in CO, D &\sqsubseteq \mathsf{Consequence} \\
\exists \mathsf{impacts}.\top &\sqsubseteq \mathsf{Consequence} \\
\top &\sqsubseteq \forall \mathsf{impacts}.\mathsf{Asset} \\
\exists \mathsf{canLeadTo}.\top &\sqsubseteq \mathsf{Consequence} \\
\top &\sqsubseteq \forall \mathsf{canLeadTo}.\mathsf{Consequence} \\
\exists \mathsf{causedBy}.\top &\sqsubseteq \mathsf{Consequence} \\
\top &\sqsubseteq \forall \mathsf{causedBy}.\mathsf{Threat} \\
\mathsf{impactedBy} &\equiv \mathsf{impacts}^{-} ,
\end{aligned}$$

where $CO = \{\mathsf{AssetDamage}, \dots, \mathsf{RegulatoryNonCompliance}\}$ is the set of consequences. As part of this process, it was necessary to remove certain individuals belonging to the Threat concept from the security ontology (e.g., Threat(assetDamage)), as the introduction of concepts that describe attack consequences rendered them superfluous. On the other hand, we added additional individuals describing generic threats, based on the threat events enumerated in the NIST SP 800-30 [59]. The ICS security ontology imports the security ontology and expands it by incorporating domain-specific knowledge that may be independent (e.g., security standards and guidelines) or dependent (e.g., tender requirements) of current engineering projects. While system-dependent knowledge has to be provided by users due to potentially changing project conditions (e.g., customers' security needs), system-independent knowledge can be modeled once and then reused for each engineering project, requiring only a minimal involvement of users from time to time (e.g., ontology maintenance tasks to incorporate revised standards). We already provide a rich system-independent ICS security knowledge model, which has been implemented with OWL and SHACL and can be directly used. This knowledge model integrates the Product, Process, and Resource (PPR) concept [53] to capture the primary types of assets modeled in AML, viz., $\mathsf{Product} \sqsubseteq \mathsf{MovableAsset}$, $\mathsf{Process} \sqsubseteq \mathsf{IntangibleAsset}$, and $\mathsf{Resource} \sqsubseteq \mathsf{ImmovableAsset}$. Further, the axiom $\mathsf{OTComponent} \sqsubseteq \mathsf{ImmovableAsset}$ is necessary to later associate the AMLsec concepts Host, NetworkDevice, and SecurityDevice as a subclass of assets concepts defined in the (ICS) security ontology. We also define the following axioms to describe the interdependencies between security and safety:

$$\begin{array}{ll}
\mathsf{SafetyAttribute} \sqsubseteq \mathsf{Attribute} & \mathsf{SafetyAttribute(availability)} \\
\mathsf{SafetyAttribute(reliability)} & \mathsf{SafetyAttribute(safety)} \\
\exists \mathsf{affects}.\top \sqsubseteq \mathsf{Threat} & \top \sqsubseteq \forall \mathsf{affects}.\mathsf{SafetyAttribute} \\
\exists \mathsf{requires}.\top \sqsubseteq \mathsf{Asset} & \top \sqsubseteq \forall \mathsf{requires}.\mathsf{SafetyAttribute} \\
\mathsf{Hazard} \sqsubseteq \mathsf{Consequence} & \mathsf{HazToEnvironment} \sqsubseteq \mathsf{Hazard} \\
\mathsf{HazToPersonnel} \sqsubseteq \mathsf{Hazard} & \mathsf{HazToResource} \sqsubseteq \mathsf{Hazard} .
\end{array}$$

Based on this, we can establish the semantic relations among assets, security threats, consequences, and security/safety attributes (i.e., protection goals that are at risk). However, the main part of this knowledge model draws upon security-relevant information from a variety of sources in order to enable the identification of vulnerabilities (and threats) in the plant topology. The knowledge modeled in OWL includes, inter alia, information about inherently insecure network protocols (e.g., Modbus), cryptographic algorithms that are considered to be insecure (e.g., SHA-1 used as part of the OPC UA security policy Basic128RSA15 [60]), ICS-focused vulnerabilities described in standards and guidelines (e.g., safety-related devices are not separated from non-safety-related devices [5]), and recommendations from vendors of industrial components (e.g., know-how protection not activated in Siemens PLCs [61]). The defined SHACL shapes, on the other hand, represent the know-how needed to identify the actual vulnerabilities in the plant topology (cf. Section 3.3.1). We opted for a SHACL-based validation to achieve a reusable and adaptable collection of rules for the purpose of identifying (ICS-specific) vulnerabilities in the plant topology. After importing the ICS security ontology into the engineering data ontology (i.e., the output of the AML-to-OWL transformation), concept equivalence axioms are added to assert the semantic equivalence of certain classes across namespaces (e.g., $\mathsf{Resource} \sqsubseteq \mathsf{AutomationMLBaseRole}$ defined in the engineering data ontology and $\mathsf{Resource} \sqsubseteq \mathsf{ImmovableAsset}$ defined in the ICS security ontology).

In accordance with the Linked Open Data (LOD) principle, the engineering and (ICS) security knowledge is interlinked with data from public sources. For example, linked information concerning vulnerabilities (e.g., CVEs) and threat activities (e.g., STIX-based alerts) can be obtained from the National Vulnerability Database (NVD) and ICS-CERT, respectively. Our method interlinks the *SEPSES Cybersecurity Knowledge Graph (KG)* [49], which is a continuously evolving source of open security knowledge integrating (i) Common Weakness Enumeration (CWE), (ii) Common Vulnerabilities and Exposures (CVE), (iii) Common Attack Pattern Enumeration and Classification (CAPEC), (iv) and the Common Vulnerability Scoring System (CVSS). We access this KG via its SPARQL Protocol and RDF Query Language (SPARQL)[3] endpoint and incorporate the result set of executed queries to establish the link to our KB.

3.3 Automated Risk Identification

Identifying security risks with our method is a two-step process that includes risk sources (i.e., threats and vulnerabilities) identification and attack consequences identification.

3.3.1 Identification of Risk Sources

The identification of risk sources rests upon the semantic relationships among vulnerabilities, assets, and threats, which are modeled in the security ontology [46]. In this context, the modeling of vulnerability relationships can be considered as a cornerstone, as it establishes the semantic links to assets ($\top \sqsubseteq \forall \mathsf{vulnerabilityOn}.\mathsf{Asset}$) and threats ($\top \sqsubseteq \forall \mathsf{exploitedBy}.\mathsf{Threat}$). Consequently, this part of our method focuses on finding vulnerabilities in assets modeled in the engineering data ontology and the subsequent

3. https://www.w3.org/TR/rdf-sparql-query

```
@prefix ... # Prefixes omitted for the sake of brevity
icsSecOnt:SafetyAssetsZone
 a sh:NodeShape ;
 sh:targetClass amlImp:Host ;
 sh:sparql [
  a sh:SPARQLConstraint ;
  sh:message "..." ;
  sh:prefixes [ ... ] ;
  sh:select """
  SELECT DISTINCT $this ?host2 ?zone1
    WHERE {
     $this a amlImp:SIS .
     ?host2 rdf:type/rdfs:subClassOf* amlImp:Host .
     ?zone1 amlOnt:hasIE+ $this, ?host2 ;
            rdf:type/rdfs:subClassOf* amlImp:Zone .
     FILTER NOT EXISTS {
      ?zone1 amlOnt:hasIE+ ?anyZone1 .
      ?anyZone1 amlOnt:hasIE+ $this ;
                rdf:type/rdfs:subClassOf* amlImp:Zone .
      }
      FILTER NOT EXISTS {
       ?zone1 amlOnt:hasIE+ ?anyZone1 .
       ?anyZone1 amlOnt:hasIE+ ?host2 ;
                 rdf:type/rdfs:subClassOf* amlImp:Zone .
      }
      FILTER NOT EXISTS {
       $this a amlImp:SIS .
       ?host2 a amlImp:SIS .
      }
    }
  """ ;
 ] .
```

Listing 2: SHACL constraint for zoning safety-related assets (checks whether only safety devices are grouped together).

creation of ontological relations using the object property vulnerabilityOnAsset. We utilize the combination of SHACL and SPARQL to achieve a powerful, yet straightforward automated vulnerability identification mechanism that makes effective use of the formalized security-relevant data. Owing to the flexibility of this mechanism, we can employ various vulnerability detection SHACL rules, from basic node and property shapes to even more complex SPARQL-based constraints. The following examples of currently implemented shapes illustrate the capabilities of our mechanism.

3.3.1.1 Node & Property Shapes: These types of shapes were used to implement validation rules that check that (i) no inherently insecure network protocols are used in modeled communication systems, (ii) no cryptographic algorithms are employed that are considered to be insecure, (iii) no unused logical endpoints exist that would unnecessarily increase the attack surface, and that (iv) PLC security functions (e.g., know-how protection) are modeled as active to ensure proper configuration.

3.3.1.2 SPARQL-based Constraints: This type was used to realize rules that check whether the following zone and conduit requirements as per the IEC 62443-3-2 [5] are met: (ZCR-3.2) separate business and control system assets, (ZCR-3.3) separate safety-related assets (cf. Listing 2), (ZCR-3.4) separate temporarily connected devices, (ZCR-3.5) separate wireless devices, and (ZCR-3.6) separate devices connected via external networks. Furthermore, we implemented a rule that identifies logical and physical connections between assets that cross zone boundaries but were not explicitly annotated with the AMLsec role class `Conduit`.

In addition, we implemented SPARQL queries requesting the SEPSES Cybersecurity KG [49] remote endpoint to determine whether CVE entries for components in the plant topology exist. If any publicly known security vulnerabilities for the components have been found, vulnerability individuals will be instantiated and the relations to the corresponding assets and threats will be established.

3.3.2 *Identification of Attack Consequences*

In the context of this work, attack consequences are the outcome of realized cyber threats that exploit one or multiple vulnerabilities and thereby negatively affect assets. Note that the attack consequences are related to security and safety attributes (e.g., availability) and therefore only capture the direct results of attacks that are caused by malicious actions performed through compromised assets. Thus, the indirect consequences resulting from cyber attacks (e.g., revenue loss, effects of reputation damage) are out of scope. In other words, we focus exclusively on identifying the types of consequences that are specific to the assets of the plant topology. To give an example, a compromised PLC controlling a pump may lead to safety issues and business interruption. Further, if the PLC has also a reference to its PLC program in AML, intellectual property theft is also identified as a potential consequence.

The automated modeling of potential attack consequences is implemented by first instantiating consequence individuals based on the interpreted semantics of plant components and then adding property relations to asset individuals. This means that we perform a mapping from the asset individuals (based on their classes) to consequence individuals. For instance, given Historian(simaticHistorian1), we can derive from our (predefined) domain knowledge that

$$\begin{gathered}\mathsf{DataBreach(dbHistorianConsequence1)}\\ \mathsf{RegulatoryNonCompliance(regHistorianConsequence1)}\\ \mathsf{impacts(dbHistorianConsequence1, simaticHistorian1)}\\ \mathsf{impacts(regHistorianConsequence1, simaticHistorian1)}\,,\end{gathered}$$

where $\mathsf{DataBreach} \sqsubseteq \mathsf{Breach}$, $\mathsf{Breach} \sqsubseteq \mathsf{Consequence}$, and $\mathsf{RegulatoryNonCompliance} \sqsubseteq \mathsf{Consequence}$. Based on the modeling of consequence individuals, we can execute SPARQL queries in order to determine which other assets (e.g., in the sense of PPR) are affected. This feature can be demonstrated with the exemplary SPARQL query shown in Listing 3 that retrieves all vulnerable assets (`?vulnDev1`, `?vulnDev2`) and the actuators (`?actuator`), such as motors, they control, through which safety issues may arise. The optional part of the query shown in Listing 3 is used to extend the result with safety-critical, vulnerable assets that maintain a logical connection to assets controlling an actuator, since we expect that some actuator control devices (e.g., robot controllers) are in turn controlled by other level 1 or level 2 devices, such as PLCs, or Human Machine Interfaces (HMIs). The results of such queries can provide valuable input for subsequent steps of the impact assessment process, in which the identified consequences are quantitatively or qualitatively estimated.

3.4 Attack Graph Generation

Research to date on AGs has provided a comprehensive set of attack modeling techniques with different characteristics

```
PREFIX ... # Prefixes omitted for the sake of brevity
SELECT DISTINCT ?actuator ?vulnDev1 ?vulnOfDev1
↪  ?vulnDev2 ?vulnOfDev2
WHERE {
 ?haz1 rdf:type/rdfs:subClassOf* icsSecOnt:Hazard .
 ?vulnDev1 secOnt:asset_impactedBy_Consequence ?haz1 ;
           secOnt:asset_has_Vulnerability ?vulnOfDev1 ;
           amlOnt:hasIE/amlOnt:hasEI ?socket1 .
 ?mWireConn a amlImp:MotorWire ;
            amlOnt:hasEI ?plug1, ?plug2 .
 ?plug1 amlOnt:hasRefPartner ?link1 .
 ?plug2 amlOnt:hasRefPartner ?link2 .
 ?socket1 amlOnt:hasRefPartner ?link1 .
 ?socket2 amlOnt:hasRefPartner ?link2 .
 FILTER ( ?plug1 != ?socket1 ) .
 FILTER ( ?plug2 != ?socket2 ) .
 ?actuator amlOnt:hasIE/amlOnt:hasEI ?socket2 .
 FILTER ( ?vulnDev1 != ?actuator ).
 OPTIONAL {
  ?logicalConn a amlImp:LogicalConnection ;
              amlOnt:hasEI ?plug3, ?plug4 .
  FILTER ( str(?plug3) < str(?plug4) ) .
  ?plug3 amlOnt:hasRefPartner ?link3 .
  ?plug4 amlOnt:hasRefPartner ?link4 .
  ?socket3 amlOnt:hasRefPartner ?link3 .
  ?socket4 amlOnt:hasRefPartner ?link4 .
  FILTER ( ?plug3 != ?socket3 ) .
  FILTER ( ?plug4 != ?socket4 ) .
  ?vulnDev1 amlOnt:hasIE/amlOnt:hasEI ?socket3 .
  ?vulnDev2 amlOnt:hasIE/amlOnt:hasEI ?socket4 .
  ?haz2 rdf:type/rdfs:subClassOf* icsSecOnt:Hazard .
  ?vulnDev2 secOnt:asset_impactedBy_Consequence ?haz2 ;
            secOnt:asset_has_Vulnerability ?vulnOfDev2 .
 }
}
```

Listing 3: Exemplary SPARQL query to retrieve all vulnerable assets that potentially cause safety issues via the controlled actuators.

and structures, most of which focus on state enumeration graphs and exploit dependency graphs or variants thereof [62], [63]. These types of graphs explicitly express the preconditions and postconditions of exploiting vulnerabilities [62] and therefore require a solid understanding of system and exploit properties for their automated generation. Although engineering data includes system information that enables the automated identification of vulnerabilities, inferring the complete preconditions and postconditions of exploits necessitates additional sources of know-how that systematically capture the prerequisites and the outcomes of successful vulnerability exploitation efforts. Moreover, we argue that users of our method would benefit from an AG that puts the vulnerable integrated components in focus rather than the network states or exploits. The rationale behind this is that a higher level of abstraction may suffice for engineers to quickly spot the most critical, cyber-physical paths adversaries may take and which vulnerable plant components are involved in these attack chains. As a result, we utilize the host-based network AG structure introduced in [39] as a basis for our cyber-physical AG, which we formally define in the following.

Definition 1. A CPAG is a directed vertex- and edge-weighted graph $CPAG = (V, E, \omega_V, \omega_E)$, where V is the finite vertex set of assets, E is a multiset of directed edges from $V \times V$ representing vulnerabilities, $\omega_V\colon V \to S$ is the vertex weight function that maps all vertices according to the assets' cyber-physical criticality onto the set S, $\omega_E\colon E \multimap S$ is the edge weight function[4] that maps all edges according to the vulnerabilities' severity onto the set S, and $S = [0, 10]$.

An edge in a CPAG, i.e., an ordered pair $e = (u, v) \in E$, exists if and only if the asset u is (logically or physically) connected to asset v and asset v has a vulnerability. Considering that assets within the plant topology can have more than one vulnerability, a CPAG may be a multigraph. Note that, in contrast to a host-based AG, $v \in V$ represents an asset within the plant topology, which does not necessarily have to be a (network) host.

For a vertex $v \in V$, the weight $\omega_V(v)$ is the potential (cyber-physical) impact on a scale of 0 to 10 of v being compromised. This allows to consider the identified types of attack consequences (e.g., safety hazards) and also accounts for the impact that may have been estimated in the risk analysis step, for example, based on PPR information (e.g., interruption of a specific production process). On the other hand, for an edge $e = (u, v) \in E$, the weight $\omega_E(e)$ is the severity on a scale of 0 to 10 of the corresponding vulnerability on v, which can be given by the CVSS score assigned to the vulnerability.

Fig. 2 shows an example of a CPAG, which comprises six assets represented as vertices $v_1 \dots v_6$, where the assets v_1 and v_2 are connected, assets v_2, v_4, and v_5 are connected, assets v_3 and v_4 are connected, assets v_4 and v_6 are connected, and v_1, v_2, v_4, v_6 have one vulnerability each (denoted by the symbols $\dagger$, $\ddagger$, $\star$, §), v_5 has two vulnerabilities (denoted by the symbols $\dagger$, $\diamond$)[5], and v_3 has no vulnerability.

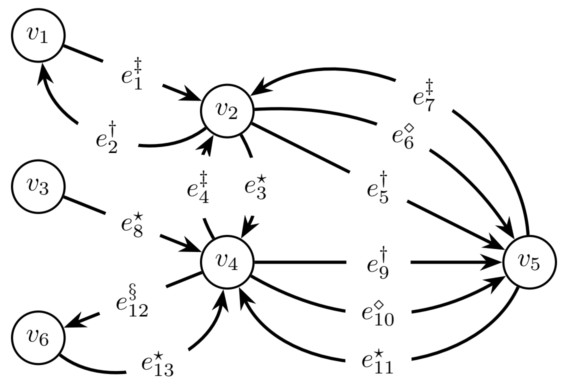


Fig. 2: Example of the introduced attack graph.

After providing a general, formal definition of CPAGs, we discuss the details of automatically generating them.

3.4.1 *Connectivity Properties*

The connectivity among assets is determined by the modeled physical and logical connections contained in engineering data. We expect that at least the topological design of the network is available in AML artifacts. Other details concerning the network architecture (e.g., segmentation configurations) stored in external documents are not considered for determining the connectivity among assets. We utilize

4. Note that ω_E is a multivalued function since E is a multiset.
5. Note that in Fig. 2, the same vulnerability $\dagger$ exists on both assets v_1 and v_5.

the concept of security zones and conduits and assume that two assets, v_1 and v_2, are physically connected to each other as long as an actual physical connection (e.g., Ethernet wire) between v_1 and v_2 has been modeled in AML and v_1 and v_2 are within the same zone. To explicitly represent these physical connections in our KB, we extend the security ontology by defining the following axioms:

$$\begin{aligned} \mathsf{phyConnectedTo} &\sqsubseteq \mathsf{connectedTo} \\ \mathsf{phyConnectedTo} \circ \mathsf{phyConnectedTo} &\sqsubseteq \mathsf{phyConnectedTo} \\ \mathsf{phyConnectedTo} &\equiv \mathsf{phyConnectedTo}^{-} . \end{aligned}$$

Note that the role phyConnectedTo is transitive, which would inadvertently result in connection relations bypassing security devices (e.g., firewalls) if the zone boundary restriction is not considered. However, connections between assets of different zones can be legitimate (conduits) and therefore must also be taken into account. Furthermore, we expect that each logical connection modeled in AML connects two assets even if security zones are crossed. Thus, we further extend the security ontology by introducing a dedicated role to describe logical connections:

$$\begin{aligned} \mathsf{logicallyConnectedTo} &\sqsubseteq \mathsf{connectedTo} \\ \mathsf{logicallyConnectedTo} &\equiv \mathsf{logicallyConnectedTo}^{-} . \end{aligned}$$

This role is used to establish connection relations between assets that are modeled in the engineering ontology with logical connection individuals of the concept LogicalConnection $\sqsubseteq$ AutomationMLBaseRole.

The connectivity properties of assets represented in the engineering ontology are set by executing SPARQL `CONSTRUCT` queries.

3.4.2 Preconditions & Postconditions

As indicated above, we relax the preconditions and postconditions of exploiting the identified vulnerabilities by making the following assumptions:

(i) The precondition for exploiting a vulnerability on asset v is that v is reachable (expressed via the physical and logical connection relations). The adversary's capabilities or other system properties (e.g., states of services running on v) are not considered.

(ii) The postcondition, resulting from the successful exploitation of the vulnerability corresponding to edge $e_i = (u, v) \in E$ on asset v, provides the means for an adversary to exploit the vulnerability corresponding to edge $e_j = (v, w) \in E$, from asset v to asset w (i.e., asset w becomes reachable).

Note that we deliberately consider also those vulnerabilities that do not elevate privilege levels upon exploitation (e.g., Vulnerability(noPLCProgramKnowHowProtection)). Yet, these vulnerabilities can be moderated by first adapting their weights accordingly and then pruning the CPAG. Furthermore, we abstain from defining initial preconditions and final postconditions (i.e., adversary's goal). However, subgraphs of a CPAG can be utilized for obtaining AGs representing specific attack goals (e.g., compromising a PLC within the control zone starting from a webserver within the business zone).

```
PREFIX ... # Prefixes omitted for the sake of brevity
CONSTRUCT {
 ?v1 a agOnt:Vertex ;
     agOnt:vertex_has_Edge ?e ;
     agOnt:vertex_has_Asset ?a1 ;
     agOnt:vertex_has_Weight ?v1W .
 ?v2 a agOnt:Vertex ;
    agOnt:vertex_has_Asset ?a2 ;
    agOnt:vertex_has_Weight ?v2W .
 ?e a agOnt:Edge ;
    agOnt:edge_has_Vulnerability ?vuln ;
    agOnt:edge_has_Vertex ?v2 .
}
WHERE {
 {
  SELECT ?v1 ?v2 ?a1 ?a2 ?e ?vuln (AVG(?v1Weight) AS
  ↪  ?v1W) (AVG(?v2Weight) AS ?v2W)
  WHERE {
  { ?a1 secOnt:asset_physicallyConnectedTo_Asset ?a2 . }
  UNION
  { ?a1 secOnt:asset_logicallyConnectedTo_Asset ?a2 . }
  FILTER ( ?a1 != ?a2 ) .
  { ?vuln secOnt:vulnerability_on_Asset ?a2 . }
  UNION {
    ?a2 amlOnt:hasIE+ ?ie .
    ?vuln secOnt:vulnerability_on_Asset ?ie .
  }
  ...
  OPTIONAL { ?a1 secOnt:asset_impactedBy_Consequence
  ↪  ?consequence1 . }
  OPTIONAL { ?a1 secOnt:asset_impactedBy_Consequence
  ↪  ?consequence2 . }
  # BIND expressions for ?v1, ?v2, ?e, ?v1Weight, and
  ↪  ?v2Weight omitted for the sake of brevity
  } GROUP BY ?v1 ?v2 ?a1 ?a2 ?e ?vuln
 }
}
```

Listing 4: SPARQL query for generating the attack graph.

3.4.3 Implementation

To formally represent the CPAG in our KB, we defined the axioms:

$$\begin{aligned} \mathsf{Vertex} \sqsubseteq\ & {=}1\mathsf{hasAsset}.\mathsf{Asset} \sqcap \exists \mathsf{hasEdge}.\mathsf{Edge} \\ & \sqcap\ \mathsf{hasWeight}.(\mathsf{xsd{:}double} \geqslant 0 \text{ and } \leqslant 10) \\ \mathsf{Edge} \sqsubseteq\ & {=}1\mathsf{hasVulnerability}.\mathsf{Vulnerability} \\ & \sqcap {=}1\mathsf{hasVertex}.\mathsf{Vertex} . \end{aligned}$$

Note that we do not introduce a role for $\omega_E(e)$, but rather reuse the role hasSeverityValue defined in the security ontology.

The automated generation of the CPAG with the introduced AG ontology is implemented by means of the SPARQL `CONSTRUCT` query shown in Listing 4. The weights of vertices are set according to predetermined criticality scores of consequence types that correspond to the vulnerable assets (`BIND` expressions omitted in Listing 4). Incorporating PPR information using subqueries is also a viable option for assigning weights to assets based on their impact on (production) processes and products.

3.4.4 Pruning

As per Definition 1, the number of edges in a CPAG is $\sum_{v \in V} con(v) vuln(v)$, where $con(v)$ is the number of assets that are connected to the asset v and $vuln(v)$ is the number of vulnerabilities on asset v. Consequently, a plant topology consisting of a considerable number of vulnerable, interconnected assets yields a substantial number of edges, thereby significantly increasing the complexity of analyzing the generated CPAG. To alleviate this issue and further improve the usability of CPAGs, a variety of pruning techniques can be applied.

For example, CPAGs can be pruned by building a subset of E based on the maximum edge weight among its duplicate elements, ensuring that only the most severe vulnerabilities remain. Furthermore, subsets of V or E can be obtained by filtering out elements whose weights are below a certain threshold value. We also argue that the consideration of typical adversarial tactics for pruning CPAGs is worthwhile. Attackers who have gained initial access to assets from the business zone may move from one asset to another until they infiltrate the control networks and achieve their ultimate attack goal. Thus, we implemented a pruning technique that removes edges directed to assets that are more distant to the control zone than their source. Similarly, edges from vertices representing assets that likely constitute final attack targets (e.g., PLCs) can be removed.

It is also worth highlighting that users can retrieve a subgraph of the generated CPAG, which can represent, for example, the shortest attack path from asset v to asset u.

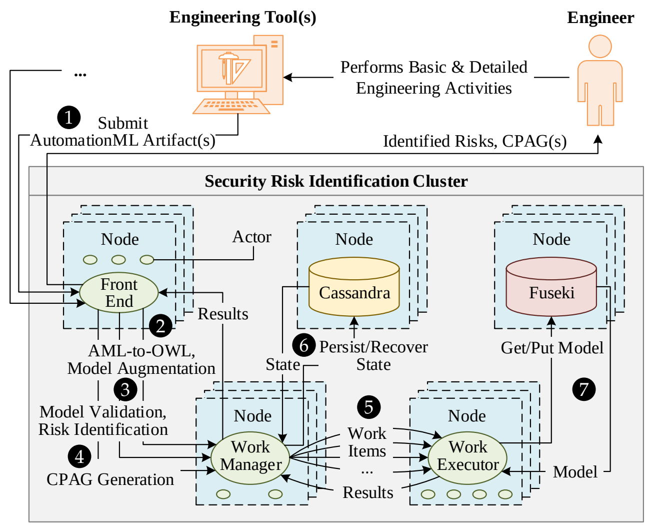

Engineering Tool(s)
Engineer
Performs Basic & Detailed Engineering Activities
...
❶ Submit AutomationML Artifact(s)
Identified Risks, CPAG(s)
Security Risk Identification Cluster
Node
Actor
Front End
Results
❷ AML-to-OWL, Model Augmentation
❸ Model Validation, Risk Identification
❹ CPAG Generation
Node
Cassandra
❻ Persist/Recover State
State
Node
Fuseki
Get/Put Model
❼
Node
Work Manager
❺
Work Items
...
Results
Node
Work Executor
Model

Fig. 3: Overview of the prototype's architecture.

3.5 Implementation

The AutomationML Editor[6] was used to create the AMLsec libraries. These libraries can be easily imported into other AML projects for reuse purposes. The ontologies were partly modeled by using the open-source ontology editor Protégé [64]. Our prototype, which executes the proposed method, was developed in Scala and utilizes the Apache Jena[7] framework and the Akka[8] toolkit. The implemented SHACL rules and SPARQL queries are reusable and additional risk identification logic can be easily added, given that our method merely depends on the semantic mapping realized by means of AMLsec. In addition to reusability and flexibility, special attention was paid to the scalability and performance of our solution. We based our prototype on Akka's implementation of the actor model [65] to build a distributed system comprising a cluster of nodes that can execute the engineering data validation and risk identification steps of our method in parallel. In this way, the system can be scaled up as engineering artifacts and risk identification rules become more and more complex.

Fig. 3 illustrates the overall architecture of the implemented prototype. Engineers export the created artifacts with their engineering tools as AML and submit these documents to a front end actor ❶. The front end actor divides the tasks of our risk identification method in work items, delegates them to the work manager, and consumes the results of each execution step. The initial work items concern the AML-to-OWL transformation and model augmentation, which are executed sequentially ❷. After that, a work item for each SHACL rule is created to validate the engineering data model and identify security risks in parallel ❸. Finally, work items for generating CPAGs are submitted ❹. All work items are received by the work manager actor, which oversees work executor actors and assigns them the tasks as they become available ❺. Furthermore, using Akka Persistence[9] with the Apache Cassandra[10] plugin, the states of an actor can be recovered (e.g., in case a node crashes) ❻. The work executor actors process the work items to perform the steps of our risk identification method and interact with the Apache Jena Fuseki[11] SPARQL server to retrieve and persist the KB ❼.

AMLsec, the source code of the prototypical implementation of our method, and the AML code used for the case study are available on GitHub[12]. Furthermore, we integrate an extended, forked version of the AML-to-OWL translation from [51], which is also available on GitHub[13].

4 Case Study

In this section, we demonstrate the benefits and feasibility of our method by means of a case study. The case study presented in this work builds upon the official AML example [52] representing a real-world robot cell of a spot welding process. This robot cell comprises four robots, four controllers, a range of peripherals, and other supplementary components required for the implementation of this process. Although this model is a realistic and self-contained representation, it lacks communication systems and the overall architecture of the plant in which this robot cell is embedded. Thus, we extended this model with architectural concepts, PPR information, and network structures, by reference to the AML white papers [54], [56] to achieve a viable engineering data representation of a realistic plant.

The architecture of the ICS modeled in AML, which will be considered for the case study, is depicted in Fig. 4. This figure illustrates the plant topology and, in particular, the resource structure of the realized PPR concept, including assets and (physical and logical) connections among them.

6. https://www.automationml.org/o.red.c/dateien.html?cat=1
7. https://jena.apache.org
8. https://akka.io
9. https://doc.akka.io/docs/akka/current/typed/index-persistence.html
10. https://cassandra.apache.org
11. https://jena.apache.org/documentation/fuseki2
12. https://github.com/sbaresearch/amlsec
13. https://github.com/sbaresearch/ETFA2019

It is worth noting that certain parts of the engineering data representation have been left out by intention in order to replicate the fragmentation and evolution of the model during the engineering process. For example, we did not populate CPE information of selected assets (e.g., `SIMATICFieldPGM6`) and limited the model fidelity in certain areas (e.g., application logic, I/O connections, network structure).

4.1 Results

To apply our method, we first supplied the AML engineering artifact containing the engineering data representation (cf. Fig. 4) as an input to our prototype to build the KB and perform the risk identification. Then, we retrieved the identified threats, vulnerabilities, and consequences from the KB. In total, our method includes 13 SHACL vulnerability rules (besides the SPARQL query for the CVE check) and automatically identified 9 threats exploiting 179 vulnerabilities in 245 assets (50 of which have the class OTComponent), which may be impacted by 43 consequences. In the following, we analyze the output of our method and discuss the utility of the obtained results (cf. Table 1 for an excerpt of the obtained findings).

The first step of the risk identification process revealed that a variety of vulnerabilities exist within the engineering data representation. Incorporating CVE information into the KB proved to be fruitful as a significant number of assets have known security vulnerabilities, making changes in the component selection or the implementation of compensating security measures necessary. Furthermore, our method correctly detected violations of the ZCR-3.2–3.6 [5]. More specifically, (i) a logical connection between `JumpSvr1` and `MES1`, which was not annotated as a conduit, prevents a clear separation of ICS assets and business or enterprise assets, (ii) safety-related assets (namely `S71510SPF1PN_1–4`) are not grouped into one zone, (iii) the engineering workstation `SIMATICFieldPGM6`, which only makes temporary connections, is not grouped into a separate zone, (iv) the wireless-enabled sensors (namely `WHARTSensor_1–3`) are not grouped into a separate zone, and (v) the remote management system `KUKAConnBox1`, which makes external connections, is not grouped into a separate zone. Our method also identified logical connections crossing zone boundaries that have not been explicitly defined as conduits, the use of insecure protocols and algorithms (`MES1` is also an OPC UA server with the security profile Basic128RSA15), and an unused logical endpoint (`SIMATICWinCCSvr1` is also a Modbus TCP/IP slave but no connected master exists).

The second step of the risk identification process exposed the consequences of compromising vulnerable assets. As can be seen in Table 1, there are several assets that adversaries may attack with the objective of causing business interruption or even safety hazards. These findings can be used to gain further insights concerning resulting consequences (e.g., the PLC `S71516F_1` is connected to KUKA robot controller `KRC4_1`, which controls the KUKA robot `030RB_100_KR240R2700prime_`). It should also not go unnoticed that our method incorporates modeled subcomponents of assets by default. For example, since the modeled plant topology includes a reference to the PLC programs running on `S71516F_1-4` our method also identified that intellectual property breach is a possible consequence of compromising these assets. Furthermore, the results regarding attack consequences can be analyzed in conjunction with PPR information by using SPARQL queries in order to assess the impact of security risks. For instance, users can determine which steps of the manufacturing process are interrupted when attacking certain assets. Estimating the physical impact based on an evaluation of which assets are indirectly affected or assessing the severity of non-compliant record keeping (e.g., loss of data integrity in pharmaceutical manufacturing) are further examples of how our method supports the impact assessment.

After performing the risk identification steps, the CPAG is generated. The full CPAG for the scenario considered in this case study contains 37 vertices and 1265 edges. Given the high number of edges, we applied the pruning techniques discussed in Section 3.4 (i.e., maximum edge weight, weight filter for vertices representing PLCs and safety-related devices[14], and filtering out atypical adversarial movement). Fig. 5 depicts the pruned version of the generated CPAG. This figure shows the possible attack paths that an adversary may follow in order to transition from the cyber to the physical domain. Based on this, users can understand how an adversary may inflict physical damages by first compromising hosts from the business zone (e.g., `FileSvr2`) and then pivoting to control devices (e.g., `KRC4_2`). In this way, users can identify, assess, and mitigate the most critical attack paths.

4.2 Discussion

Showcasing the utility of our method in a case study naturally raises the question how well our contribution generalizes across different plant models and how much effort is involved in applying the developed concepts in other scenarios.

A fundamental part of the presented method is the semantic mapping between the engineering data representation and the security ontologies. We assume that the engineering data already exists in AML, and that engineers annotate the model with certain security-relevant information (see step ❶ in Fig. 1). Based on this, semantic relations among the engineering data and the security know-how (e.g., via the definition of concept equivalence axioms) are established. The following modeling concepts support users in applying the presented method and make it generalizable:

- *Direct use of AML community resources:* We utilize a subset of the basic AML role classes and interface classes from [54], [55], [56] to harness community resources that are already widely adopted in the industry. Obviously, additional base classes (e.g., from domain-specific libraries that are shared among organizations) can be incorporated.

14. To focus on a specific part of the spot welding process at hand, we increased the weights of the vertices corresponding to `S71516F_2`, `KRC4_2`, and `SimaticS71510SPF1PN2`, thereby filtering out other control devices.

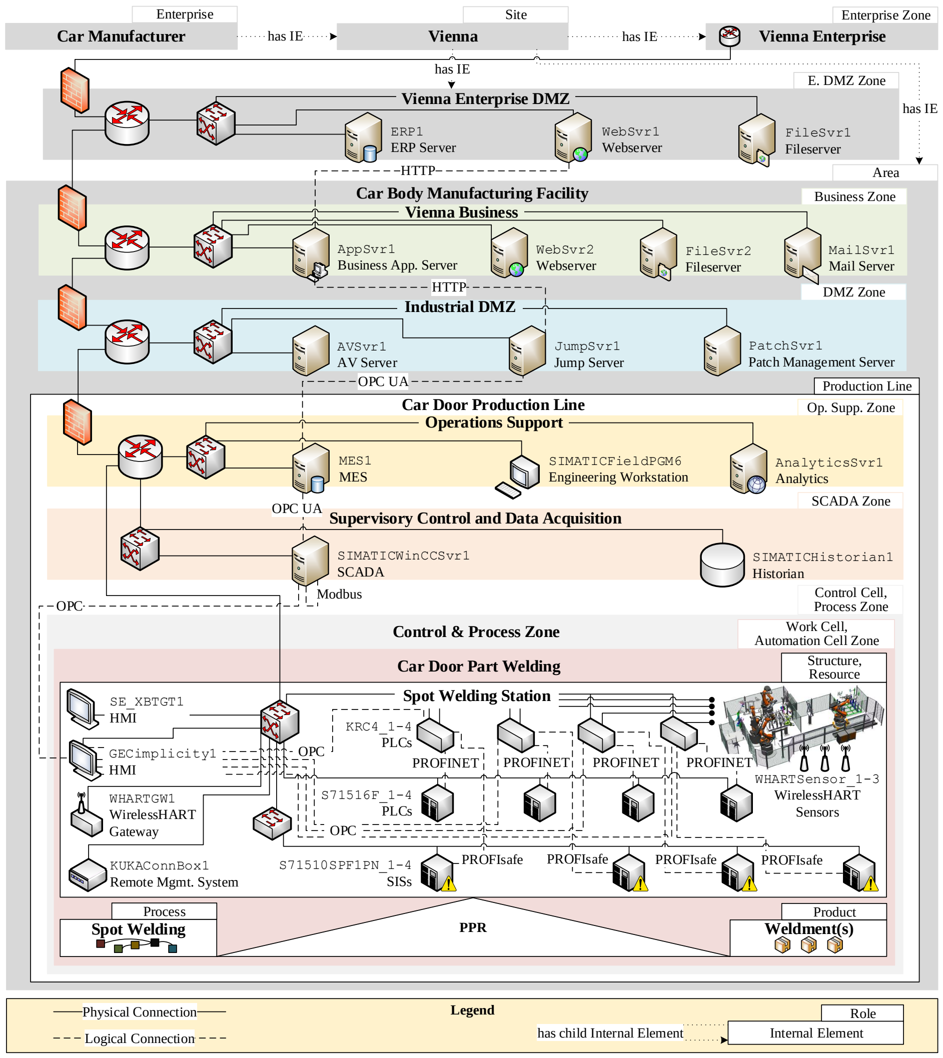


Fig. 4: Illustrated plant topology modeled in AutomationML (robot cell illustration taken from [52]).

Asset(s)	Vulnerabilities										Consequences				
	CVE	ZCR-3.2	ZCR-3.3	ZCR-3.4	ZCR-3.5	ZCR-3.6	CON	PRO	ALG	ULE	BI	DB	IP	HAZ	REG
ERP1	●	-	-	-	-	-	-	-	-	-	●	-	-	-	-
WebSvr1	●	-	-	-	-	-	●	●	-	-	-	-	-	-	-
AppSvr1	●	-	-	-	-	-	●	●	-	-	-	-	-	-	-
WebSvr2	●	-	-	-	-	-	-	-	-	-	-	-	-	-	-
JumpSvr1	-	●	-	-	-	-	●	●	-	-	-	-	-	-	-
MES1	●	●	-	-	-	-	●	-	●	-	●	-	-	-	-
SIMATICFieldPGM6	-	-	-	●	-	-	-	-	-	-	-	-	-	-	-
SIMATICWinCCSvr1	●	-	-	-	-	-	●	-	-	●	●	-	-	●	-
SIMATICHistorian1	●	-	-	-	-	-	-	-	-	-	-	●	-	-	●
SE_XBTGT1	●	-	-	-	-	-	-	-	-	-	●	-	-	●	-
GECimplicity1	●	-	-	-	-	-	●	●	-	-	●	-	-	●	-
WHARTSensor_1-3	-	-	-	-	●	-	-	-	-	-	-	-	-	-	-
KUKAConnBox1	-	-	-	-	-	●	-	-	-	-	-	-	-	-	-
S71516F_1-4	●	-	-	-	-	-	-	●	-	-	●	-	●	●	-
S71510SPF1PN_1-4	●	-	●	-	-	-	-	●	-	-	●	-	-	●	-
KRC4_1-4	-	-	-	-	-	-	-	●	-	-	●	-	-	●	-

Column descriptors: CVE: Common Vulnerabilities and Exposures entry exists, ZCR-3.2–3.6: Zone and Conduit Requirements according to [5], CON: cross-zone connection not marked as conduit, PRO: insecure protocol, ALG: insecure algorithm, ULE: unused logical endpoint, BI: business interruption, DB: data breach, IP: intellectual property breach, HAZ: hazards (safety risks), REG: regulatory non-compliance.

TABLE 1: Excerpt of the results of the automated vulnerability and consequence identification, indicating which assets in the considered plant topology (cf. Fig. 4) are vulnerable and the corresponding attack consequences.

- *AMLsec:* We provide three libraries comprising several security-relevant AML role classes, interface classes, and attribute types. These libraries are versatile, can be easily imported into existing AML files, and extended if needed, making them an ideal complement to the AML base libraries.

All steps subsequent to modeling, from AML-to-OWL transformation to CPAG generation, are fully automated. Furthermore, both the presented AML semantic constructs and the provided security rules are domain independent, and can be adapted as necessary. While adapting the rules is straightforward, establishing entire rule sets as community resources would be an option for the future to achieve a wide coverage and ensure that they remain up to date.

5 Performance Evaluation

To assess the performance and scalability of our prototype we conducted a series of tests with multiple input models which vary in size. Table 2 provides an overview of the datasets A–F, which were used for the assessment. Dataset A corresponds to the input model which was used in the case study, comprising an enterprise with only one site. The datasets B–F include an enterprise with two, four, six, eight, and ten sites (each of which is a copy of the Vienna `InternalElement`), respectively.

We measured the prototype's execution time in five experiments that we performed for each dataset with two cluster configurations (i.e., 60 runs in total). The first cluster consists of three nodes. One node runs the front end actor, the work manager actor, the Apache Jena Fuseki service, and the Apache Cassandra service. The remaining two nodes host work executor actors. For the second setup, we created a five-node cluster that has two additional work executor nodes. In both clusters, each work executor node runs three actors. All nodes are cloud-hosted virtual machines running Fedora 32 x64 with 8 vCPUs (based on the Intel® Xeon® Gold 6140 processor) and 16 GB RAM.

It is also worth mentioning that we materialize the inferences made by the reasoner as part of the model augmentation step during the setup phase in order to improve the performance at the cost of a moderate increase of the KB size (cf. Table 2). Moreover, we executed the tests with Jena's OWL micro reasoner[15], which is not as powerful as the default one in terms of the supported constructs but in turn achieves a higher performance. Using the micro reasoner instead of the default reasoner has only a minimal effect on the functionality of the implemented method, as the not supported `owl:disjointWith` constructs are merely used for validity checks involving disjoint (union) relations (cf. Section 3.2), which can be compensated by creating SHACL rules if needed.

5.1 Results

Fig. 6 shows the results of the performance assessment. In Fig. 6a, we illustrate the average execution time of important steps of the setup phase (i.e., AML-to-OWL transformation, model augmentation) and the CPAG generation phase that we collected from tests with both cluster configurations (i.e., 10 runs for each dataset), since the steps of these phases are not executed by multiple actors in parallel. In contrast, Fig. 6b and Fig. 6c show the average execution time separately for both clusters (2, and 4 work executor nodes).

15. https://jena.apache.org/documentation/inference

	A	B	C	D	E	F
Engineering Data						
`InternalElements` (in K)	0.56	1.12	2.24	3.36	4.47	5.59
AML Size (in MB)	0.56	1.30	2.60	3.90	5.10	6.40
After AML-to-OWL Trans.						
Triples (in K)	9.47	18.17	35.56	52.94	70.33	87.72
KB Size (in MB)	1.20	2.30	4.50	6.80	9.00	11.20
After Method Execution						
Triples (in K)	39.65	76.61	167.73	281.78	418.75	578.66
KB Size (in MB)	4.30	8.40	18.30	30.30	44.50	60.80
Assets (in K)	0.25	0.49	0.98	1.47	1.95	2.44

TABLE 2: Overview of the datasets used for the evaluation.

In the following, we report the average execution time and the standard deviations for the largest dataset F, which contains 2441 assets. The execution time of the AML-to-OWL transformation and model augmentation for dataset F averaged 96.79 ± 10.62 seconds and 114.41 ± 28.68 seconds, respectively. The optional model validation by the reasoner averaged 126.83 ± 93.25 seconds for the largest dataset (and 1.02 ± 0.21 seconds for the smallest dataset). Validating the model (with our SHACL rules) and identifying security risks for the largest dataset averaged 1151.09 ± 522.00 seconds and 695.62 ± 43.03 seconds with two, and four work executor nodes, respectively. The average execution time for generating a full CPAG for the largest dataset is 32.44 ± 3.12 seconds.

Pruning the full CPAG afterwards to obtain the graph shown in Fig. 5 averaged 10.48 ± 0.99 seconds and 76.09 ± 9.28 seconds for dataset A and F (smallest and largest dataset), respectively. The total execution time of our method (from the setup phase to pruning the CPAG as presented in the case study, including miscellaneous implementation-related tasks) with the three-node cluster averaged 57.95 ± 9.54 seconds and 1729.60 ± 330.14 seconds for datasets A and F, respectively. On the other hand, the total execution time obtained with the five-node cluster averaged 58.18 ± 2.08 seconds and 1280.44 ± 139.24 seconds for datasets A and F, respectively.

5.2 Discussion

Our method builds upon a set of SPARQL queries and SHACL rules; hence, its performance depends primarily on the implementation of the W3C SPARQL/SHACL standards and reasoners, the rules/queries to execute, and the underlying structure and size of the KB. We have demonstrated that scaling out provides significant performance gains, as the load of the method execution can be spread across multiple nodes. For instance, from Fig. 6b it is evident that with two additional work executor nodes, the average execution time of the security risk identification phase decreases dramatically and the sample has lower variability, implying a more consistent performance as extreme values are less likely. The reason for this is that the work items are randomly assigned among nodes, potentially leading to

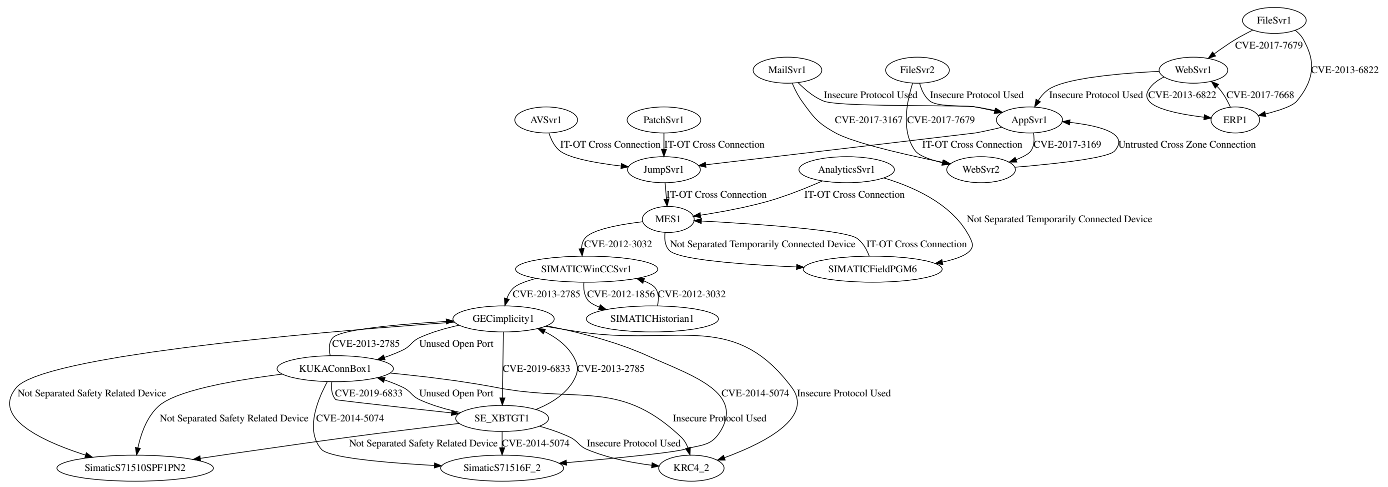


Fig. 5: Pruned cyber-physical attack graph that was automatically generated for the case study.

the case that long-running tasks are unevenly distributed. Consequently, we suggest to add more work executor nodes to the cluster as the datasets become larger or more security rules are added to the KB. Additional performance considerations apply to the setup phase. Materializing the inferences during the setup phase reduces the load on the work executor nodes. Furthermore, since the setup phase only needs to be performed once for each engineering data representation, both the security risk identification and CPAG generation can be rerun without needing a fresh model setup, if new security know-how becomes available at a later point in time. Possibly, the performance can be further enhanced by excluding non-security-relevant individuals as part of the AML-to-OWL transformation to optimize the KB for a faster execution of certain triple patterns in queries (e.g., property paths). Lastly, we want to point out that engineers can check their models periodically (e.g., at certain project milestones) or continuously throughout the engineering process to receive timely feedback. For instant results (e.g., providing live feedback in the engineering tool), it would be worthwhile to check only those parts of the model that include recent changes instead of the full version.

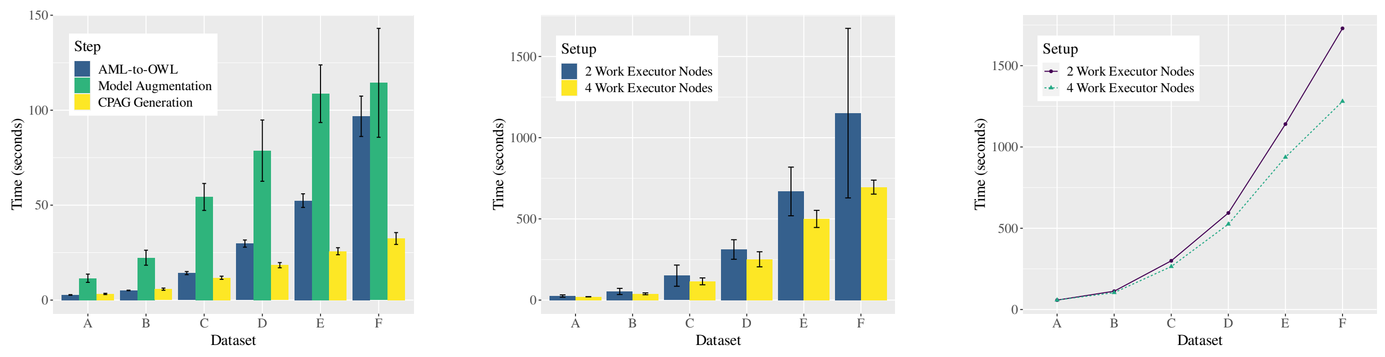
6 Conclusion

In this work, we have presented a novel method that automatically identifies sources of security risks and the corresponding attack consequences based on engineering data. We build our approach on engineering data described in AML, i.e., a standardized format widely used in the engineering domain for data exchange. The introduction of AML extension libraries named AMLsec equips AML with security semantics, while ensuring a seamless integration into the engineering workflow and minimizing additional modeling efforts. AML documents are first transformed into an ontological representation and then enriched with additional security know-how, allowing to employ semantic reasoners to derive new security-relevant knowledge and to automate the risk identification. In this context, we developed a comprehensive set of SHACL constraints and SPARQL queries to obtain security risks from the knowledge base in an automated manner, thereby supporting security risk assessments according to IEC 62443-3-2 [5]. Furthermore, we have proposed a new variant of AGs named CPAG that is specifically geared to the needs of systems integrators to understand potential lateral movement attacks against the engineered CPSs and their physical manifestation. The automated generation of CPAGs, including the developed pruning techniques, ensure that this attack modeling technique can unfold its full potential to support engineers in designing more secure CPSs. The capabilities of our method and the implemented open-source prototype are demonstrated in a case study. It is evident from the results provided that our method unburdens individuals from risk identification by automating tedious, effortful, and knowledge-driven tasks, allowing them to focus on other steps of the risk management process. The presented performance assessment shows that the proposed risk identification method is feasible, even when dealing with considerable large datasets. Owing to the distributed nature of the prototype, the security risk identification can be executed in parallel by multiple nodes, resulting in improved performance and greater scalability. Consequently, our prototype can be seamlessly embedded into the engineering workflow to execute the security risk identification on demand or as part of a pipeline in the engineering chain.

Finally, we want to provide pointers to future work. We plan to investigate how the ICS security ontology can be managed in a sustainable and efficient way. In this context, mining industrial security standards and guidelines, and establishing the system-independent security know-how as a collaborative public source, are research directions worth pursuing. Ideally, in the future, our method is also capable of automatically performing (quantitative) security risk analyses and subsequent treatment. Furthermore, there is still room for improvement to further boost the performance of our prototype. For instance, distributing long-running rules more evenly and executing certain tasks of the setup phase in parallel is certainly worth implementing.

Acknowledgment

The authors thank Magdalena Ortiz, Georg Merzdovnik, Ludwig Kampel, and Holger Knublauch for their valuable



(a) Setup and CPAG generation (b) Validation and risk identification (c) Total

Fig. 6: Performance assessment results of our implemented prototype (error bars indicate standard deviations).

comments and suggestions.

The COMET center SBA Research (SBA-K1) is funded within the framework of COMET — Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG. Moreover, the financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

[1] P. Kieseberg and E. Weippl, "Security challenges in cyber-physical production systems," in *Software Quality: Methods and Tools for Better Software and Systems*, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Cham: Springer International Publishing, 2018, pp. 3–16.

[2] M. Eckhart, A. Ekelhart, A. Lüder, S. Biffl, and E. Weippl, "Security development lifecycle for cyber-physical production systems," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, Oct 2019, pp. 3004–3011.

[3] R. Drath, A. Luder, J. Peschke, and L. Hundt, "AutomationML - the glue for seamless automation engineering," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Sept 2008, pp. 616–623.

[4] S. Faltinski, O. Niggemann, N. Moriz, and A. Mankowski, "AutomationML: From data exchange to system planning and simulation," in *2012 IEEE International Conference on Industrial Technology*, March 2012, pp. 378–383.

[5] IEC, "62443-3-2: Security for industrial automation and control systems – part 3-2: Security risk assessment and system design," *International Standard, Draft, International Electrotechnical Commission, Geneva*, vol. 1, 2018.

[6] M. Eckhart, B. Brenner, A. Ekelhart, and E. Weippl, "Quantitative security risk assessment for industrial control systems: Research opportunities and challenges," *Journal of Internet Services and Information Security (JISIS)*, vol. 9, no. 3, pp. 52–73, Aug. 2019.

[7] M. Rocchetto, A. Ferrari, and V. Senni, *Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems*. Cham: Springer International Publishing, 2019, pp. 25–47.

[8] N. Schmidt and A. Lüder, "AutomationML in a nutshell," AutomationML e.V., techreport, Nov. 2015.

[9] IEC 62424, "Representation of process control engineering – requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools," *International Standard, Second Edition, International Electrotechnical Commission, Geneva*, vol. 2, 2016.

[10] ISO 31000:2009, "Risk management – Principles and guidelines," International Organization for Standardization, Geneva, CH, Standard, Nov. 2009.

[11] M. Glawe, C. Tebbe, A. Fay, and K.-H. Niemann, "Knowledge-based engineering of automation systems using ontologies and engineering data," in *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, ser. IC3K 2015. Portugal: SCITEPRESS - Science and Technology Publications, Lda, 2015, pp. 291–300.

[12] C. Tebbe, M. Glawe, A. Scholz, K.-H. Niemann, A. Fay, and J. Dittgen, "Wissensbasierte Sicherheitsanalyse in der Automation," *atp magazin*, vol. 57, no. 04, pp. 56–66, 2015.

[13] M. Glawe and A. Fay, "Wissensbasiertes Engineering automatisierter Anlagen unter Verwendung von AutomationML und OWL," *at-Automatisierungstechnik*, vol. 64, no. 3, pp. 186–198, 2016.

[14] C. Tebbe, M. Glawe, K.-H. Niemann, and A. Fay, "Informationsbedarf für automatische IT-Sicherheitsanalysen automatisierungstechnischer Anlagen," *at-Automatisierungstechnik*, vol. 65, no. 1, pp. 87–97, 2017.

[15] S. Fluchs and H. Rudolph, "Wie OT Security Engineering eine Ingenieurwissenschaft wird," *atp magazin*, vol. 61, no. 8, pp. 74–86, 2019.

[16] ——, "Making OT security engineering deserve its name," *Control Global*, Nov. 2019.

[17] S. Fluchs, N. Schmidt, and A. Mendl-Heinisch, "Ein Systemmodell für Security-Engineering," *Technische Sicherheit*, vol. 11-12/2019, pp. 35–44, Dec. 2019.

[18] J. Jürjens, "UMLsec: Extending UML for secure systems development," in *«UML» 2002 — The Unified Modeling Language*, J.-M. Jézéquel, H. Hussmann, and S. Cook, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 412–425.

[19] ——, *Secure Systems Development with UML*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[20] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *«UML» 2002 — The Unified Modeling Language*, J.-M. Jézéquel, H. Hussmann, and S. Cook, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 426–441.

[21] M. J. M. Chowdhury, "Security risk modelling using SecureUML," in *16th Int'l Conf. Computer and Information Technology*, March 2014, pp. 420–425.

[22] D. A. Robles-Ramirez, P. J. Escamilla-Ambrosio, and T. Tryfonas, "IoTsec: UML extension for internet of things systems security modelling," in *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, Nov 2017, pp. 151–156.

[23] L. Apvrille and Y. Roudier, "SysML-Sec: A SysML environment for the design and development of secure embedded systems," in *APCOSEC 2013, Asia-Pacific Council on Systems Engineering, September 8-11, 2013, Yokohama, Japan*, Yokohama, Japan, 09 2013.

[24] R. Oates, F. Thom, and G. Herries, "Security-aware, model-based systems engineering with SysML," in *Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research 2013*, ser. ICS-CSR 2013. UK: BCS, 2013, pp. 78–87.

[25] L. Lemaire, J. Lapon, B. De Decker, and V. Naessens, "A SysML extension for security analysis of industrial control systems," in *Proceedings of the 2Nd International Symposium on ICS & SCADA Cyber Security Research 2014*, ser. ICS-CSR 2014. UK: BCS, 2014, pp. 1–9.

[26] J. Wittocx, M. Mariën, and M. Denecker, "The IDP system: A model expansion system for an extension of classical logic," in *Proceedings of the 2nd Workshop on Logic and Search*, Denecker, Marc. ACCO; Leuven, 2008, pp. 153–165.

[27] L. Lemaire, J. Vossaert, J. Jansen, and V. Naessens, "Extracting vulnerabilities in industrial control systems using a knowledge-based system," in *Proceedings of the 3rd International Symposium for*

ICS & SCADA Cyber Security Research, ser. ICS-CSR '15. Swindon, UK: BCS Learning & Development Ltd., 2015, pp. 1–10.

[28] T. Sommestad, M. Ekstedt, and H. Holm, "The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures," *IEEE Systems Journal*, vol. 7, no. 3, pp. 363–373, Sept 2013.

[29] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt, "P^{2}CySeMoL: Predictive, probabilistic cyber security modeling language," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 626–639, Nov 2015.

[30] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar, *Probabilistic Relational Models*. MIT Press, 2007, pp. 129–174.

[31] T. Sommestad, M. Ekstedt, and P. Johnson, "A probabilistic relational model for security risk analysis," *Computers & Security*, vol. 29, no. 6, pp. 659 – 679, 2010.

[32] S. Kriaa, M. Bouissou, and Y. Laarouchi, "A model based approach for SCADA safety and security joint modelling: S-cube," *IET Conference Proceedings*, pp. 6 .–6 .(1), January 2015.

[33] M. Bouissou, H. Bouhadana, M. Bannelier, and N. Villatte, "Knowledge modelling and reliability processing: Presentation of the figaro language and associated tools," *IFAC Proceedings Volumes*, vol. 24, no. 13, pp. 69 – 75, 1991, iFAC Symposium on Safety of Computer Control Systems 1991 (SAFECOMP'91), Trondheim, Norway, 30 October-1 November 1991.

[34] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Computer Science Review*, vol. 13-14, pp. 1 – 38, 2014.

[35] Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby, and K. Stoddart, "A review of cyber security risk assessment methods for SCADA systems," *Computers & Security*, vol. 56, pp. 1 – 27, 2016.

[36] K. Kaynar, "A taxonomy for attack graph generation and usage in network security," *Journal of Information Security and Applications*, vol. 29, pp. 27–56, 2016.

[37] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders, "Adversary-driven state-based system security evaluation," in *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, ser. MetriSec '10. New York, NY, USA: Association for Computing Machinery, 2010.

[38] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke, "Model-based security metrics using adversary view security evaluation (ADVISE)," in *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, Sep. 2011, pp. 191–200.

[39] S. Zhong, D. Yan, and C. Liu, "Automatic generation of host-based network attack graph," in *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 1, March 2009, pp. 93–98.

[40] P. Ammann, J. Pamula, R. Ritchey, and J. Street, "A host-based approach to network attack chaining analysis," in *21st Annual Computer Security Applications Conference (ACSAC'05)*, Dec 2005, pp. 10 pp.–84.

[41] A. Xie, G. Chen, Y. Wang, Z. Chen, and J. Hu, "A new method to generate attack graphs," in *2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement*, July 2009, pp. 401–406.

[42] S. Wu, Y. Zhang, and W. Cao, "Network security assessment using a semantic reasoning and graph based approach," *Computers & Electrical Engineering*, vol. 64, pp. 96 – 109, 2017.

[43] K. Falodiya and M. L. Das, "Security vulnerability analysis using ontology-based attack graphs," in *2017 14th IEEE India Council International Conference (INDICON)*, Dec 2017, pp. 1–5.

[44] J. Lee, D. Moon, I. Kim, and Y. Lee, "A semantic approach to improving machine readability of a large-scale attack graph," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3028–3045, Jun 2019.

[45] E. F. Hill, *Jess in Action: Java Rule-Based Systems*. USA: Manning Publications Co., 2003.

[46] S. Fenz and A. Ekelhart, "Formalizing information security knowledge," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 183–194.

[47] C. Tebbe, K.-H. Niemann, and A. Fay, "Ontology and life cycle of knowledge for ICS security assessments," in *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016*, ser. ICS-CSR '16. Swindon, UK: BCS Learning & Development Ltd., 2016, pp. 1–10.

[48] J. Wolf, F. Wiezorek, F. Schiller, G. Hansch, N. Wiedermann, and M. Hutle, "Adaptive modelling for security analysis of networked control systems," in *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016*, ser. ICS-CSR '16. Swindon, UK: BCS Learning & Development Ltd., 2016, pp. 1–10.

[49] E. Kiesling, A. Ekelhart, K. Kurniawan, and F. Ekaputra, "The SEPSES knowledge graph: An integrated resource for cybersecurity," in *The Semantic Web – ISWC 2019*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, Eds. Cham: Springer International Publishing, 2019, pp. 198–214.

[50] Y. Hua and B. Hein, "Concept learning in AutomationML with formal semantics and inductive logic programming," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, Aug 2018, pp. 1542–1547.

[51] ——, "Interpreting owl complex classes in automationml based on bidirectional translation," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 79–86.

[52] AutomationML, "AutomationML example: Robot cell," Tech. Rep., Mar. 2017.

[53] M. Schleipen and R. Drath, "Three-view-concept for modeling process or manufacturing plants with automationml," in *2009 IEEE Conference on Emerging Technologies Factory Automation*, Sep. 2009, pp. 1–4.

[54] AutomationML, "Whitepaper AutomationML edition 2.1 part 1 — architecture and general requirements," Tech. Rep. V 2.1.0, Jul. 2018.

[55] ——, "Whitepaper AutomationML part 2 — role class libraries," Tech. Rep. V 2.0.0, Oct. 2014.

[56] ——, "Whitepaper AutomationML communication," Tech. Rep. V 1.0.0, Sep. 2014.

[57] IEC 62714-1, "Engineering data exchange format for use in industrial automation systems engineering – automation markup language – part 1: Architecture and general requirements," *International Standard, Second Edition, International Electrotechnical Commission, Geneva*, vol. 2, 2018.

[58] L. Abele, C. Legat, S. Grimm, and A. W. Müller, "Ontology-based validation of plant models," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, July 2013, pp. 236–241.

[59] Joint Task Force Transformation Initiative, "Guide for conducting risk assessments," *NIST Special Publication*, vol. 800, no. 30r1, Sep 2012.

[60] M. Gallinat, S. Hausmann, M. Köster, and S. Heiss, "OPC-UA: Ein kritischer Vergleich der IT-Sicherheitsoptionen," in *KommA 2014 — Jahreskolloquium Kommunikation in der Automation*, Lemgo, Germany, Nov 2014.

[61] Siemens, "SIMATIC S7-1500 security: Getting started," Siemens, Tech. Rep. A5E03982396-01, Mar. 2013.

[62] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Computer Science Review*, vol. 35, p. 100219, 2020.

[63] M. A. Alhomidi and M. J. Reed, "Attack graphs representations," in *2012 4th Computer Science and Electronic Engineering Conference (CEEC)*, Sep. 2012, pp. 83–88.

[64] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen, "Creating semantic web contents with Protégé-2000," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 60–71, March 2001.

[65] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular ACTOR formalism for artificial intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, ser. IJCAI'73. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1973, pp. 235—-245.

Matthias Eckhart is researcher at SBA Research and the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI).

Andreas Ekelhart leads the department for applied research projects at SBA Research and is senior researcher at the Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle (CDL-SQI).

Edgar Weippl is full professor at the University of Vienna, co-founder and Research Director of SBA Research, and the Head of the Christian Doppler Research Laboratory Security and Quality Improvement in the Production System Lifecycle (CDL-SQI).