Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems

Alberto Ferrari[1], Marco Rocchetto[1], and Valerio Senni[1]

[1]United Technology Research Center (**name.surname@utrc.utc.com**)

Abstract The design of Cyber-Physical Systems (CPS) poses a number of challenges, in particular for cyber-security. Eliciting Security Requirements is a key aspect in the early system design stages; however it is important to assess which requirements are more stringent and grant protection against the higher-value assets. Cyber-security Risk Assessment (SecRA) has a key role in determining threat scenarios and evaluating the risks associated to them but it is a practice that has been principally developed for IT systems, thus focusing on cyber threats. In this chapter, we discuss the state of the art in SecRA methodologies and the challenges to be addressed for developing new CPS-oriented SecRA methodologies. Based on the most relevant standards for industrial control systems and automotive domain (such as the ISA/IEC-62443 and the J3061), we propose the adoption of an asset-driven viewpoint and a model-based approach to SecRA, and we identify current gaps. In particular we discuss (i) CPS (security) modeling languages and methodologies, (ii) vulnerabilities cost models and the network of public repositories of vulnerabilities, (iii) attacker models and profiles, and (iv) complex cyber-physical attack chains. Finally, we discuss our vision, focusing on assets and leveraging model-based design practices can provide a more rigorous approach to SecRA for CPS, allow taking into consideration their peculiarities, and support to manage the large complexi-

ty involved in their operation. The desired outcome is to provide the system design team with methods and tools to identify complex attacks and perform a cost/benefit tradeoff analysis to justify the adoption of specific Security Requirements and the necessary costs implied by the corresponding mitigations.

1. Introduction

The design of modern cyber-physical, embedded systems poses complex challenges in terms of cyber-security. The adoption of (1) more complex and distributed electronics (e.g. in the automotive domain [1]), (2) an increasing number of software components (e.g. as reported in [2] the Airbus A320 avionics system has around 80000 lines of code while the Boeing 777 exceeds 4 million lines), and (3) remote monitoring, configuration, maintenance, and software update capabilities (e.g. firmware update over-the-air [3]) provide new entry-points to security threats. Therefore, system designers need to face the complexity of securing a significantly increased system attack surface. Following a secure-by-design approach [4], several standards provide guidelines and processes to support the design and deployment of secure systems by careful evaluation of external system entry-points and internal architecture from the perspective of an attacker (e.g. ISA/IEC-62443 for Industrial Control Systems [5], the ISO/IEC-27000 family [6] and NIST 800-53 for IT Security [7], DO-356 for Avionics Industry [8], and J3061 for the automotive domain [9]). A key step, identified in the processes proposed by standards, is that of *Cyber-security Risk Assessment (SecRA)*, which supports the identification of threats and the evaluation of the risks to which the system is exposed. The principal outcome of SecRA is a set of *mitigations* for the higher criticality risks, which are translated into *derived security requirements* and added to a set of Minimum Security Requirements (MSR) either (1) obtained directly from the customer or (2) product/domain-specific, or (3) inherited from regulations and standards (such as the common criteria of ISO/IEC-15408 [10]). See Figure 1 for a conceptual workflow showing the role of Risk Assessment and compare also to the J3061 process [9].

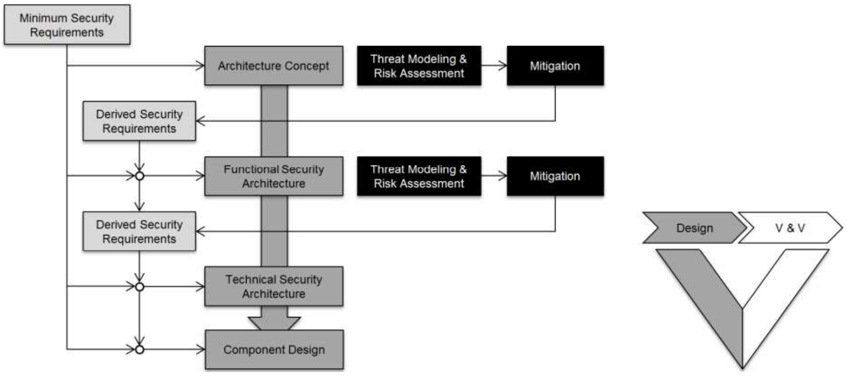
Minimum Security Requirements
Architecture Concept
Threat Modeling & Risk Assessment
Mitigation
Derived Security Requirements
Functional Security Architecture
Threat Modeling & Risk Assessment
Mitigation
Derived Security Requirements
Technical Security Architecture
Component Design
Design
V & V

Figure 1. Role of Risk Assessment in a Secure-by-Design workflow.

SecRA is typically performed as a structured process [11] that guides through the identification of the key system *assets*, the elicitation of relevant *threats* and of the vulnerabilities they can leverage, and an informal evaluation of the *impact* (with an associated *likelihood)* that those threats can have on the system assets. Knowledge of both the application domain and of the (expected) system design is critical to ensure an effective evaluation of the risks.

In this chapter, we discuss two potential shortcomings of this common approach to SecRA. First, the reasoning behind the analysis of impact and likelihood is often not supported by formal artifacts (e.g. functional/architectural diagrams [12] [13] or attack trees [14]). Thus, the evaluation of *completeness of the assessment and correctness of the results relies* entirely on the system engineers (and/or security experts), with the side effect of producing *few or no artifacts that document the rationale behind the analysis* (e.g., for future and third-party reviews). Second, the classic SecRA process provides limited support for the *analysis of the impact on risk evaluations of any system design change*, due to the absence of the aforementioned architectural artifacts and consequent lack of traceability of risks to system design elements (e.g. to architecture elements).

The role of SecRA is well understood for IT/software systems (and a number of tools exist to support it [15]), but this is not the case for Cyber-Physical Systems (CPS), which are complex systems involving software, hardware, actuators, sensors, plant and environment interactions with strict performance and safety constraints [16]. The challenge of secure design of these systems stems from the complex interaction between physical and virtual components, exposing the physical world to the effects of cyber threats, with direct impacts on safety (e.g., Stuxnet [17] and other examples in [18]). Therefore, the aforementioned shortcomings of common SecRA approaches are more relevant for CPS and cannot be overlooked.

Our *first claim* is that we envision a fruitful synergy between Model-Based Design ([19]) and SecRA resulting in an “MB-SecRA” approach that can (1) improve confidence on completeness and correctness of SecRA results by providing reviewable artifacts documenting the risk assessment rationale, and (2) provide support to change management by explicit traceability of risks to design elements.

A second challenge, rising in modern CPS, is the increasing complexity of hardware and software, which opens CPS to the risk of sophisticated *multi-step attacks*. A notable example is the aerospace domain; with a tremendous increase in use of software to replace hardware functionalities (e.g. see the study on software complexity from NASA [20]). Similarly, in the automotive domain, the number of lines of code (LOC) and of CPUs present in a modern car makes the system highly subject to advanced cyber-attacks [9]. In commercial applications, to reduce the time to market, it is common to reuse existing software components, protocols or platforms and this raises the concern of how wide is the impact of a known vulnerability. There are large public databases such as OWASP [21] or CWE [22] that report hundreds of vulnerabilities affecting widely adopted software components. For hardware components there are no databases of equal scope yet but we envision a growing need in this area. This complexity of the software/hardware architecture opens up for complex attacks known as *advanced persistent threats* (APT), where an attacker (willing to invest resources and time to archive its objectives) is able to reach core system assets by exploiting multiple (apparently not so critical) vulnerabilities in a synergistic way (*kill- or attack-chains* [23]) to achieve increasingly more information about the system and higher privileges, until he is able to produce system-level damages. Kill chains and APTs are very hard to identify at system design time and also at run-time due to their sophistication [24].

For a correct evaluation of cyber-security risks in complex systems it is necessary to change the traditional approach of considering the impact of a single vulnerability in isolation and turn the attention to attack chains. From this perspective, a vulnerability may be discovered to play a critical role in multiple attack chains and thus become one of the highest-ranked among the identified mitigation actions.

A growing number of Security Risk Assessment tools and methodologies (e.g., CORAS [25], ThreatModeler [26], Microsoft STRIDE [27]) adopt design artifacts such as software/hardware architecture schemes, network topology diagrams, and data-flow diagrams to support the evaluation of system-level effects of local vulnerabilities. However, they leave the actual evaluation of the impact and the risk caused by those vulnerabilities to a manual and informal analysis of those artifacts performed by a system engineer. This direction seems promising but still suffers from two issues: first, *the number of vulnerabilities to be combined can give rise to hundreds of potential attack scenarios* and a high review complexity, and second, there is *limited documentation of the reasoning behind the analysis of system-level impacts of local vulnerabilities*. Our second claim is that, the contribution of MBD into Cyber-security Risk Assessment should not be limited to driving the identification and evaluation of risks (e.g., by means of formal architectural artifacts) but also leverage abstract behavioral models that allow the representation of data- and function-flows that an attacker can use to propagate the effects of a vulnerability exploitation. A successful approach in the design of high-assurance systems has been the adoption of formal, automated and exhaustive analysis methodologies to ensure the absence of undesired behaviors in software design (e.g. see adoption of formal methods in avionics [28]), including absence of cyber-security vulnerabilities such as potential attacks in protocols design [29]. We believe formal security analysis can be successfully applied also in the area of Risk Assessment.

The objectives of this chapter are (1) to identify challenges and opportunities to improve current SecRA methodologies for the specifics of CPS, and lay the basis of an MB-SecRA approach to improve confidence on completeness and correctness of risk assessment by leveraging formal and traceable model-based artifacts and (2) provide a high-level a workflow for formal analysis of known vulnerabilities and identification of attack-chains, and finally, (3) to discuss several open challenges and gaps that need to be filled to realize the proposed MB-SecRA approach.

Structure

In Section 2, we discuss the state-of-the-art of SecRA and the open challenges for CPS. In Section 3, we discuss the opportunities we envision in developing a model-based approach to system, attacker and vulnerabilities modeling for Risk Assessment. In Section 4, we describe a possible roadmap to implement model-based SecRA and use a small example to illustrate the concepts.

2. Background and Open Challenges

Cyber-security Risk Assessment has been extensively discussed over the past years [30] [31] and attracted special attention in the field of SCADA systems [32] given their role in managing and controlling critical infrastructures. Despite the large amount of work in the field, this area is in practice still widely addressed by using informal artifacts and tools [33] (such as Excel spreadsheets) and strongly relies on the domain expertise of review teams.

In this section, we discuss the open challenges that we identified for the application of model-based SecRA for CPS. Moreover, we discuss the opportunities to increase the effectiveness of SecRA in terms of (1) completeness and correctness of the results, (2) capability of managing design changes, and (3) understanding of system-level effects of attack-chains.

2.1 Cyber-security Risk Assessment Methodologies

Following to the ISO 31000 [31] (a consolidated standard providing a framework for risk management), Risk Assessment is characterized by three main activities: (1) Risk Identification, (2) Risk Analysis, and (3) Risk Evaluation.

In the cyber-security context [30], Risk Identification is the process of recognizing and describing risks. Based on the *Assets* (what is protected), the *Incidents* (events that have negative consequences on the Assets), and the system cyber-security *Vulnerabilities* (design or implementation flaws that, if exploited, can cause an Incident), a *Risk* identifies the conditions under which external or internal *Threats* can exploit existing vulnerabilities with the purpose of causing an incident and, thus, a damage to the Assets. The Risk Analysis activity has the objective to review the identified risks and to provide a quantitative estimate for the *likelihood* of a specific risk and the related *impact* on assets. Finally, during the Risk Evaluation activity, each risk is compared with the evaluation criteria. The quantitative estimate of impact and likelihood is used to determine the risks that should be considered for treatment. Risk treatment typically implies the identification of derived security requirements indicating the required security measures, e.g. elimination of vulnerabilities or restriction of accessibility. Several methodologies for SecRA leverage a representation of the system architecture and data-flow to support a more formal and repeatable approach to Risk Identification and Risk Analysis.

Microsoft proposes a methodology based on (1) the threat classification model STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privileges) and (2) a language for threat modeling based on abstract software elements (Entities, Processes, Data-flows, Data Stores, Trust Boundaries) to represent and analyze paths that links vulnerabilities to incidents [34]. STRIDE than allows to choose to proceed either "per component" or "per interaction" to perform the assessment of risks. A tool (Microsoft Threat Modeler) implements STRIDE "per component" and supports the application of STRIDE categories to each component in the threat model. Pre-determined templates are available for typical software components (e.g. services, databases) with specific questions to characterize the risk for the specific component. In this methodology the architecture model has mainly the role of supporting reviews and ensuring completeness of SecRA, but does not allow understanding the actual effect at system-level of the component vulnerabilities. This focus on a specific library of software components together with the lack of system-level assessment leads us to consider this methodology not adequate to CPS.

The SESAR SecRAM methodology [35] has an implicit dependency to the notion of system architecture. It starts from the identification of *Primary Assets*, that are abstract critical system functions or services, and adopts an approach similar to STRIDE by considering a categorization of the threats based on the CIA (Confidentiality, Integrity and Availability) paradigm. Based on these categories, SecRAM is able to provide a preliminary assessment of the impact areas and a corresponding evaluation of criticality. By leveraging a representation of the physical architecture of the system, SecRAM maps the Primary Assets to the *Physical Assets*: impacts are inherited in this mapping process. Thus, the methodology can proceed backwards considering the vulnerabilities of each component and producing chains Vulnerability-Impact-Asset. The final step consists in computing the likelihood of the risks. The advantage of SecRAM is to take into account the system-level by starting from Primary Assets. The mapping to the (physical) architecture provides a better understanding of how the vulnerabilities affecting single (physical) components might impact on Primary (Immaterial) Assets. Still, this valuable information is provided by the system and security engineers through an informal review.

CORAS [25] [36] is an explicitly model-driven SecRA methodology in the sense that models are not implicit but are adopted to support and execute all the Risk Identification, Analysis and Evaluation activities. The diagrams and views provided by CORAS are designed to be straightforward and to enable the capture and documentation of relations between threats, vulnerabilities, incidents and impact on assets. The diagrams are created with the purpose of supporting discussion and documenting *structured brainstorming sessions*. CORAS is the methodology closer to our vision since its process is based on a functional justification of the vulnerabilities effects. Threat Diagrams are graphs structured into specific layers: (1) a mapping of Assets to Incidents affecting them, (2) a mapping of Incidents to a chain of internal System States and Vulnerabilities enabling this chain, and (3) a mapping of vulnerabilities to Threats that can exploit them. This representation provides a valuable representation of the potential path of an attack (Threat Scenario), it is useful to justify how a Threat agent is expected to exploit Vulnerabilities to affect the Assets, and clarifies the risk analysis rationale. However, Threat Diagrams are very high-level and applicable for the concept design phase [12] (i.e., the very first engineering design step). They are created in a non-rigorous way by system and security engineers and their level of abstraction makes it hard to extract concrete attacks from the results (such as those obtained from CWE [22], NVD [37] or threat reports). The motivation is to be found in the fact that Threat Diagrams are unrelated to the actual system logical/physical architecture and behavior. Thus, there is no formal justification for the identified Threat Scenarios, captured and described by the analyst based on his experience.

Our claim is that the manual extraction of Threat Scenarios is extremely complex and highly error prone when considering a CPS, and may be unfeasible for large-scale CPS, where there can be thousands of complex attacks. For this reason we see the opportunity of improving state-of-the-art risk assessment approaches to allow the systematic extraction of Threat Scenarios by leveraging different system viewpoints (e.g., logical and physical architectural views, behavioral and vulnerability models, attacker models). Models can also support automated extraction of Threat Scenarios, thus making the Risk Assessment activity less error prone. In the context of model-based system design flows ([19]), systematic extraction of Threat Scenarios from models can pave the way to the identification of potential attacks that are easier to reproduce on the actual system in the security validation phase.

2.2 CPS Design Languages for Security

The systematic or automated extraction of Threat Scenarios and the consequential analysis of risks are applicable under the assumption that the CPS design is based on models and a specific security viewpoint is captured and documented. In this section, we briefly review the state-of-the-art of system modeling languages and their support for security viewpoints.

CORAS is based on UML [13], a robust and highly adopted [38] (open) standard modeling language for software engineering. In UML, it is possible to define two main types of diagrams: structural (e.g., class and package diagrams), and behavioral (e.g., sequence and activity diagrams). The *structural diagrams* are used to decompose software into different parts (e.g., packages, classes, methods) while the *behavioral diagrams* supports the design of the semantics of those different parts. UML supports mechanisms, called profiles, which allow its users to make extensions to the language itself (i.e., semantics refinements of UML). The CORAS language for risk modeling was initially defined as a UML profile [39] [40], and standardized as part of the UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (UML QoS&FT). There exist several other profiles that extend UML to model security properties or risk-related information. Examples of the former are SecureUML [41] and UMLSec [42], while examples of the latter are Abuse-Cases by McDermott and Fox [43].

In [43], McDermott and Fox first proposed the idea of applying specialized use-cases for the purpose of threat identification, and misuse-cases [44] (that extends UML use-cases) to elicit security requirements. Another approach is provided by UMLSec, that extends UML with tags (called stereotypes) that allow the modeler to define security properties and constraints. Similarly, SecureUML extends UML allowing the modeler to express access control policies as constraints. The focus of

UML is mainly on software development rather than embedded systems or CPS. On the other hand, SysML [45] (Systems Modeling Language) extends UML to support the design of complex hardware/software systems by introducing new diagrams to define requirements and constrains on the system properties (e.g., performance or reliability) and block diagrams to better describe the structure of hardware/software components and interfaces. Similarly to UML, SysML extensions have been proposed with the goal of introducing security concepts to the SysML language. SysML-Sec [46] [47], developed in the context of the EU project EVITA [48], is one such extension. SysML-Sec is designed to take into account security and safety during the engineering development of embedded systems and shares some commonalities with our vision. Specifically, SysML-Sec leverages SysML block and state machine diagrams enriched with security aspects that allow the modeler to consider both architectural and behavioral aspects. However, SysML-Sec lacks of any physical aspect in attack modeling and no support for considering attack-chains exploiting multiple vulnerabilities in sequence.

The Architecture Analysis and Design Language (AADL) is designed for the specification, analysis, automated integration and code-generation of real-time distributed systems [49]. In [50], the authors consider a set of security requirements that have been proved to mitigate set of attacks (exploiting known weaknesses in the software architecture designs) to authentication and input validation methods. The authors were able to define architecture design constraints that, if satisfied, ensure the satisfaction of those highly relevant security requirements. Exploiting this result and the available analysis tools for AADL they were able to formally prove whether an architecture is robust against attacks to authentication and input validation mechanisms. The approach leverages STRIDE to derive specific authentication and validation requirements for the system under analysis, then these requirements are used to validate the architecture against the above mentioned security requirements. This approach provides high assurance against a fixed set of security requirements that can be expressed in terms of architectural constraints. The challenge we propose is to develop an approach that is not specific for a set of security requirements but can be applied to general security requirements to validate both the system architecture and the system behavior.

2.3 Open Challenges

In this section, after the review and assessment of the state-of-the-art, we summarize the challenges identified for Cyber-security Risk Assessment of CPS:

1. *Model-driven method for Cyber-security Risk Assessment of CPS.* The adoption of models can (a) increase confidence and completeness in risk assessment, (b) provide formal support for a more objective evaluation and documentation of the risk assessment rationale through reviewable artifacts, (c) support change management though traceability of risks to design elements.
2. *Support for risk assessment of complex cyber-physical attacks*. To overcome the complexity of modern CPS hardware and software, the high number of vulnerabilities, and to be able to evaluate risks of sophisticated multi-step attacks we envision the development of tools that can automate portions of the risk assessment leveraging the models fostered in challenge (1). The research

community is well aware of this problem and has been actively working on that over the past few years ([51] [52] [53] [54] to name a few).

3. *Formal models of CPS*. Modelling a CPS is challenging because it requires to consider physical aspects (e.g. accessibility) together with software and network aspects. Further understanding of potential attacks can be achieved considering also the system behavior, that is, use cases, data flows and components' role in the protection of the system assets. For this reason it is important to develop a view-based approach to system modeling to allow easier maintenance and review of the different viewpoints.
4. *Formal models of vulnerabilities and attackers*. To have a formal approach to assessing complex risk scenarios it is necessary to capture vulnerabilities and their effects on the nominal system behavior. There is still no broadly accepted library of vulnerabilities even though (as we are going to describe in Section 3.3.2) there exists a number of libraries of public vulnerabilities. The problem with existing vulnerability libraries is twofold: (a) representation of vulnerabilities is often informal, and (b) for industrial products it is often impossible to retrieve vulnerability reports. For this reason, a vulnerability library is an important asset for a company and should be devised to be reusable to mitigate maintenance costs. A second important part of the risk assessment model is the attacker, which is required to capture the potential interactions with the system as well as the cost and likelihood of the single events.

3. Model-Driven Cyber-security Risk Assessment

In this section, we discuss some opportunities we currently identified in the roadmap to address the open challenges discussed so far. We consider, in particular, abstraction levels, expressiveness, and complexity of CPS modeling. We propose an approach to vulnerabilities and attackers modeling and we also consider aspects related to cost models.

3.1 CPS Formal Model and Abstraction Level

The adoption of formal system modeling languages is influenced by *expressiveness*, *usability*, and support by *automated tools*. Expressiveness of a formal language is directly related to the *level of abstraction* of the system model. Modeling the system in high details has the advantage of lowering the gap between the model and the real system, resulting in more precise analyses, at the cost of maintenance and high skills required to develop the models. The correct tradeoff between complexity and abstraction level is the principal aspect to consider. However, for security there is also another axis to be considered, which is the complexity of defining *correct security models and properties*, see for example the interesting discussion on how to define a system to be secure [55]. To give an example of this, a vulnerability may be publicly available and studied, but the impact of that vulnerability has to be related to the specific system and depends also from the perception of the system owner. For this reason Assets are a key input that should be provided by the system owner for any security analysis.

To evaluate the system-level impact of a vulnerability, a model of a CPS should consider both its *architecture* and its *behavior*. The architecture captures the *topology* and the *interactions* between components; while the behavior defines the *dynamics* and *functionalities* of the CPS. The formal modeling of architectures has a fairly extensive reference literature and several surveys exist on the topic (see, e.g., [56] for the protocols, and [57] for the architecture). Formal models of protocols behavior typically rely on transition system expressing how the information is sent (structure of the packets) and the evolution of the knowledge of the parties involved in the communication. Similarly, the architecture should take into account how the topology allows the exchange of information between subsystems. The behavior of the overall system relies on the behaviors of the various components of the system and, in turn, on the inputs/outputs generated and sent between components.

As discussed in [51], the correct modeling of the physical environment (e.g., the dynamic of the system or the laws of physics) of the CPS plays an important role in the correctness of the modeling and of the results of the security analysis. To mitigate the effort and complexity of modeling the physical environment there are opportunities in automated identification of the dynamic of the system [58], which is still an open research challenge. It is important to consider that faithful physical models may be hard to analyze formally [59] and can benefit from domain specific abstractions leveraging expressive formal languages [60]. Core formal analysis engines (e.g., Z3 [61], nlSAT [62], Yices [63], MathSAT [64], CVC4 [65]) have made big steps forward in solving non-linear mathematical models, thus making formal analyses closer to be applicable to CPS, see for example the promising benchmarks recently obtained by the NuXMV model checker [66]. In summary, there are several open scientific challenges in security analysis of CPS and we see opportunities emerging for formal and automated extraction of attack scenarios.

3.2 Attacker Models

In the literature there is an extensive list of (formal or semi-formal) attacker models. One of the most widespread attacker models is the so called Dolev-Yao model (DY) [67]. The DY has been extensively used in the past few decades in a number of security protocol verification tools (e.g., ProVerif [56], MaudeNPA [68]). For the purpose of security analyses, protocols are modeled as a set of agents exchanging messages over a network. The (DY) attacker is usually assumed to be part of the network and to be able to read as well as modify the messages, performing operations such as encryption and concatenation. Protocol analysis typically applies the 'perfect security' assumption, where the attacker cannot break cryptographic but only leverage the protocol logic for the attack.

The model of the attacker for CPS shall be different from the ones considered for protocol verification [53]. Usually, the focus is not on cryptography but more on the control of the network extended with some physical properties (e.g., the physical location of the attacker with respect to the CPS [69]). In fact, the physical part of the CPS allows the attacker to perform a number of attacks which extends to physical interactions with the system (e.g., physical-layer interactions or side-channel attacks). There is still no unified theory of a cyber-physical attacker model but some approaches have proposed their attacker model for CPS.

Profiling the Attacker

We remark that the focus of risk assessment is not on identifying new vulnerabilities but rather to be able to precisely estimate the risk associated to known vulnerabilities. Therefore, we are not interested in an attacker that can show, e.g., new flaws on a CPS but on an attacker that can leverage known vulnerabilities to impact the assets of a CPS.

According to [53] "an *Attacker Model* (together with compatible system models) will ideally fully characterize the possible interactions between the attacker and the system under attack. In particular, the model will define constraints for the attacker (e.g. finite computational resources, no access to shared keys)". Below we summarize the main characteristics of the attacker (i.e., attacker profile) that can be used as a basis for the definition of a vulnerability model. In the description we stress in italics the metrics we derived from [53].

- *Knowledge*. We consider the worst-case scenario and a *fair* attack surface, where the attacker has an incomplete understanding of the system under attack (*system*). The model specifies sub-system that are not directly accessible from the attacker, but also the knowledge of the attacker about those sub-systems and how they work (e.g. what communication protocols are in place, *credentials*, etc.). The attacker has (partial) visibility over a protocol or the functioning of a component but, in general, cannot directly modify the behavior of components (*source code*). Therefore, we can consider cases where the attacker just waits until the system reaches a specific configuration so that he can perform his attacks. The *offensive* skills of the attacker are strongly connected to the vulnerabilities of the system, i.e., an attacker can interact with the system as a regular user (*physical*), and can exploit attacks that leverage vulnerabilities of the system.

- *Resources*. We consider scenarios where the attacker has physical access to the system, and scenarios where the attacker needs to exploit some vulnerability to access to the system. The *effort* that the attacker puts into his attacks is also an important parameter. It is possible to formalize the notion of cost for each attack step and have an analysis to take into account the cost model for effective analysis of the most likely attack.
- *Psychology.* The attacker is *dishonest* and driven by (i) the maximization of the impact of a vulnerability on the nominal behavior of the system and on his knowledge, and (ii) the cost of exploiting such vulnerability (*aim*). The *strategy* of the attacker is driven by the exploitation cost of the vulnerabilities and the assets of the system model.

The attacker model can be seen as an extra component that is able to stimulate both the nominal inputs of the system and also the vulnerabilities, which are able to change the nominal behavior of the system. We also envision having another element of the model that is able to capture costs and impacts. The attacker may choose to exploit a vulnerability based on the cost associated to the relative attack. In Figure 2, we provide a conceptual representation of how a system design model can be extended with a model of the attacker and a model of costs/impacts.

The key aspects that we have not discussed yet are how to model vulnerabilities, their effects on the components behavior, and the cost of exploitation.

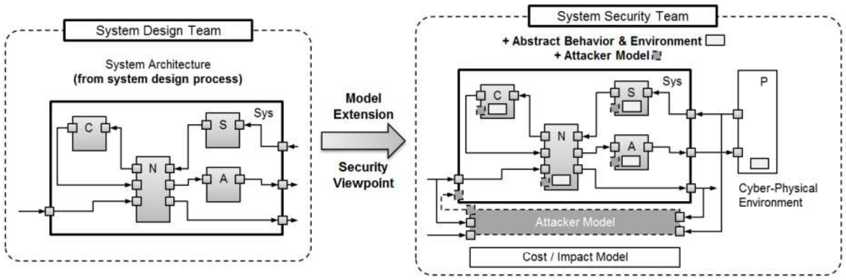

Figure 2. System Design extended with a Security Viewpoint.

3.3 Vulnerability and Cost Models

Formal risk assessment requires correct modeling of vulnerabilities and their effects **[70]**. Typically, public vulnerability repositories do not provide a description of vulnerabilities effects that is sufficiently detailed to support *formal* modeling. This is not surprising, because vulnerabilities effects are software/hardware/OS-dependent and are much better described in terms of the mechanism that they exploit, to allow covering several cases and conveying the logic of the attack, rather than the technical detail. Therefore, a model of the vulnerability effect is an effort that is application-specific and becomes an important asset of a company. To mitigate this effort it is important to define ad-hoc libraries (which often remains private) **[71]** that enable reuse. Another approach is to encode vulnerabilities in the system models as it is done, for example, in security mutation testing **[72]**. In order to provide a general methodology, similarly to **[71]**, we now discuss the opportunities available in public software vulnerability repositories.

The National Vulnerability Database [37] (NVD), is a public database provided by the National Institute of Standards and Technology of the U.S. Department of Commerce. NVD provides a detailed overview of each vulnerability by providing

(1) an informal description of the attack vector and the conditions under which the vulnerability is exploitable, (2) affected software and versions, and (3) a scoring number, defined according to the Common Vulnerability Scoring System (CVSS) [73] and providing an evaluation of the attack complexity and of the potential impacts. Every NVD description has a unique Common Vulnerability and Exposures (CVE) identifier and description [74]. The CVE database is maintained by the MITRE organization with the objective of facilitating and standardizing information exchange on vulnerabilities.

The Open Vulnerability and Assessment Language (OVAL) [75] is an important tool for automated assessment of vulnerabilities: it provides a formal description of the necessary preconditions for exploitation of a CVE entry. OVAL descriptions are structured to be processed by an automated tool in order to evaluate whether a CVE is applicable to a specific system/environment.

Another important source of information is the database of Common Weaknesses Enumerations (CWE) [22]. A CWE entry describes a weakness that can occurs in a software architecture, design, code or implementation that can lead to exploitable security vulnerabilities. So a CWE description provides details and examples on poor software designs that can lead a software system to be subject to a CVE. Some CVE entries are linked to one or more CWE entries.

Patterns of use of vulnerabilities are captured in the Common Attack Pattern Enumeration and Classification (CAPEC™) [76] database, where attack prerequisites, outcomes, indicators, execution flow, severity, solutions and mitigations, attacker skills or knowledge required, and attack variations are captured. Used together, CWE and CAPEC provide a complete viewpoint on where and how software is likely to be attacked.

The amount of information provided by this network of repositories covers several important areas for understanding risks in software design: (i) typical attack patterns and vulnerabilities they exploit, (2) existing preconditions on software design, (ii) the cost of exploitation and access of a vulnerability, and (iii) the typical impacts of the vulnerabilities. An open challenge is how to create a similar infrastructure for *cyber-physical* systems. The principal limitation we should overcome is lack of disclosure of information on commercial HW/SW components.

3.3.1 Mitigation/Exploitation Cost of Vulnerability Exploits

We consider two different costs: the cost for the attacker to exploit an attack due to a vulnerability of the system (*exploitation cost*), and the cost to mitigate/fix the vulnerability and prevent the attacks associated to it (*mitigation cost*). Information on the exploitation cost could be derived from the vulnerability databases described in the previous section. The estimate of the mitigation cost is depending from a number of factors that are out of scope for the current discussion, including the estimation of the value of the asset to be protected.

We now discuss metrics provided by the CVSS system and can be used to define an estimate of the exploitation cost of an attack:

- *Severity base score*, estimates the severity of a CVE. An attacker is likely to put more effort in exploiting high-severity vulnerabilities. The severity of the CVSS (version 3.0) is divided into four categories: low (0.0-3.9), medium (4.0-6.9), high (7.0-8.9), critical (9.0-10.0). This value is calculated as a function of other metrics: exploitability, scope, and impact.

- *Exploitability score*, is the most important factor in the equation to calculate the exploitation cost. It estimates how easy it is for an attacker to exploit a vulnerability through an attack. The CVSS exploitability score relies on the following metrics.
    - *Attack vector*, the context by which the exploitation is possible.
    - *Attack complexity*, considers the conditions, beyond the attacker control, that must exist to exploit the vulnerability.
    - *Privileges required*, determines the level of privileges an attacker must have to exploit the vulnerability.
    - *User interaction*, determines if the vulnerability requires user involvement or collaboration to be exploited.
- *Scope (or authorization scope).* A Boolean value used to estimates if a vulnerability can impact resources beyond its means of privileges.
- *Impact score*. Estimates the impact on the core confidentiality, integrity and availability security properties.

Considering cyber-physical systems, there is a number of areas where research on the definition of adequate metrics. For example, we should include the physical distribution of the system, the physical accessibility, the effort to influence the state of a plant and disguise supervisory controls. On the other side, the potential impacts are enormous and therefore be motivating for higher investment.

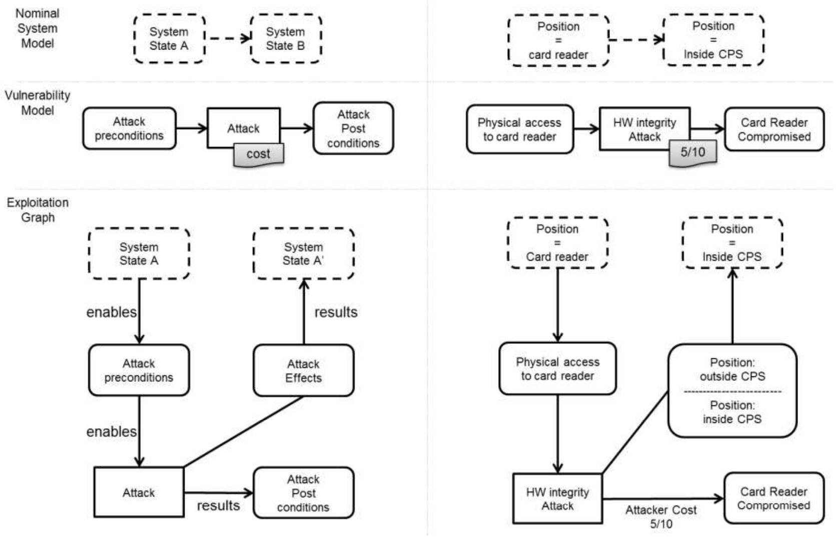

Figure 3 (left) Exploitation graph and (right) its implementation in a toy example

3.3.2 A High-level Representation of Vulnerabilities

A first approximation for representing vulnerabilities for the purpose of high-level Security Risk Assessment is to describe (1) the preconditions that allow the exploitation of the vulnerability by the attacker, and (2) how it impacts the system. An exploit can be used whenever the preconditions are in place in the current status of the system. However, the attacker does not use all the exploits whenever he can, but he considers the cost of using an exploit and tries to select the cheapest combination of exploits that reaches the attacker's goals. The impact of the exploitation of vulnerabilities in CPS is a deviation of the system behavior from the nominal one, leading to damages to the system assets.

If we assume the nominal system behavior is represented as a transition system an exploit can be modeled as an enriched transition from a nominal state of the system model to a new state, enabled by the exploitation of a vulnerability. As depicted in Figure 3 (left), a vulnerability is enabled when specific attack preconditions are satisfied in a system state. The vulnerability, once exploited, results in a set of post-conditions, whose effects entail a variation of the system state. Whenever a vulnerability exploit is executed, the attack cost is incremented and a new system state is reached.

As an example, in Figure 3 (right), we assume that an access control panel requires demonstrating valid credentials through use of a card to access to a restricted maintenance room of a CPS. The vulnerability is modeled so that whenever the attacker has physical access to the authentication panel (precondition), he can exploit an integrity vulnerability (attack), e.g. by physical tampering of the card reader, and gain access to the CPS. The post-conditions are expressed in terms of the effects of the exploitation of the attack, i.e., the changing of the physical position of the attacker. This is a deviation from the system nominal behavior that expects the access to be granted only to persons that own a badge.

4 A Vision for CPS Security Risk Assessment

We now provide a summary of the challenges and opportunities discussed so far in the form of a high-level description of how we envision SecRA to be performed for CPS. The purpose is to stimulate further discussions and to serve as a starting point for the definition of a research roadmap on this novel and challenging area. The summary we propose leverages practices that are already adopted in safety-relevant and high-assurance systems (e.g. model-based design flows [19], structured approaches to security and safety from the automotive and aerospace domains [9] [8], and standards for Cyber-security Risk Assessment [5]) but requires to address multiple challenges before reaching the required level of automation and formalism.

1. *Secure system architecture and behavior design* is structured into progressive refinement steps, supported by models, and organized into viewpoints:

    1.2 *Refinement steps:* (ref. to Figure 1) the design starts from the definition of a *concept*, where there is no notion of security and the principal system functions are identified and allocated to a high-level physical architecture, then is refined into a *functional security architecture,* where (given a decomposition of system functions into component functions and a refinement of the physical architecture by defining interfaces and data-flows) the security measures and controls are identified on the basis of a preliminary SecRA, then it is finalized into a *technical security architecture*, where (given a complete definition of system functions, physical components interfaces and data-flows) the security measures and controls are defined in details on the basis of the results of SecRA.

    1.3 *Models:* should cover both the architecture and the behavior

        1.3.1 *Architecture:* shall be used to understand the attack surface, the attack paths, the location of security controls and measures, the layers of security to design defense in depth.

        1.3.2 *Behavior:* shall be modeled to support automated extraction of threat scenarios, to support simulation of identified threat scenarios, and also to allow better run-time security measures and controls.

1.4 *Viewpoints:* shall be used to decompose the system modeling activity into teams (function, safety, security, validation, environment, etc.), to manage complexity of the overall design, and to leverage work shared across team.

2. *Adoption of Formal Behavioral Models:* the adoption of formal models can be challenging, require specific expertise, and cause an extra effort in the process. However, it opens up the opportunity of performing automated and exhaustive analyses, where the provided assurance is higher than with other methods. It requires the use of specific modeling languages, supported by formal analysis tools, that are typically very effective to model digital systems and have currently partial support for complex, physics-based models. For all these reasons, we envision the use of formal and automated analyses for SecRA either at the level of the functional security architecture, where more abstract models can be used, or for the Risk Assessment of specific high-criticality components.

3. *Compositionality and Reuse:* the architecture and behavioral models should be developed in a compositional approach (ref. to Figure 2), to allow replacement and reuse of different models to construct multiple versions of the *Nominal System Model*, defining the system behavior under normal conditions. Compositionality applies also to the *Attacker Model* since, depending on the asset we should protect and on the environment the system shall operate in, we envision reusing the nominal system behavior model, and composing it with different attacker profiles. Similar considerations apply to the *Cost and Impact Model* and, clearly to the *Environment Model*. Finally, we envision the need of defining and maintain a *Library of Vulnerability Models* that should be applicable

by composition to the single components and have the role of changing the nominal component model to a *Component Under Attack Model*. This component-driven extension approach shall support the derivation of a *System Under Attack Model* without additional effort.

4. *Security Goals and Risk Assessment:* we take an asset-driven approach to SecRA, which (in our experience) allows us to better define the security goals in terms of minimizing the loss of value of the assets. Security Goals should be formulated in terms of description of events that have an impact on the assets (*Incidents*) and should be avoided. Asset values can be used to provide a quantitative ranking of the Incidents. Therefore, formal analysis can be used to evaluate multiple scenarios that can lead to an Incident by leveraging known vulnerabilities of the system. A *cost/benefit analysis* can provide a ranking of the different attacks, in terms of the cost they require from the attacker and the impact they can cause on Assets. From this analysis, we expect to extract the most critical risks and perform a root-cause analysis to determine what vulnerabilities are more influential and shall be mitigated.

As discussed in Section 2.2, there are several existing languages and toolchains that can support the System Architecture and Behavior Design step described previously. A notable example is SysML, which embeds the notion of viewpoint natively, supports compositional definition of behaviors by Sequence Diagrams or Statecharts [77], and is implemented in several toolchains, that typically allow exporting models to perform analyses leveraging external tools [78].

Concerning the modeling of component vulnerabilities, as described in the Compositionality and Reuse step, there are several challenges and opportunities, described in Section 3.3. Our current vision on how to extend the system nominal behavior has been described in Section 3.3.2. The effect of a vulnerability (which can be extracted from public repositories such as NVD) is expressed in terms of behavioral changes of the nominal system model or as changes to the knowledge of the attacker. The latter case can lead to a reduction of the cost associated to the exploitation of vulnerabilities, thus influencing the Cost/Impact Model.

Finally, there is an interesting aspect to be discussed on the Security Goals and Risk Assessment step. Security Goals should capture conditions that avoid Incidents to Assets. Our current approach on this aspect is to formally define Security Goals as invariants with an associated numeric value and a related Asset. Violation of an invariant represents a state of the system where an Incident occurred and damage has been caused to an Asset with a specific value loss. This allows ranking attacks in terms of the value loss they produce. Let us now consider an example where some of the ideas discussed so far can be seen in practice.

Example. We consider a Water Treatment Plant (WTP) as in Figure 4, completely disconnected from the Internet, during operation, located in a Machine Room physically accessible through a Maintenance Access Gate through an Authentication Panel (AP). The WTP is periodically maintained by a technician, who connects his laptop to the Supervisory Control PC to collect performance and diagnosis data, which are sent back to the WTP production company, both for diagnosis purposes and for performance analysis.

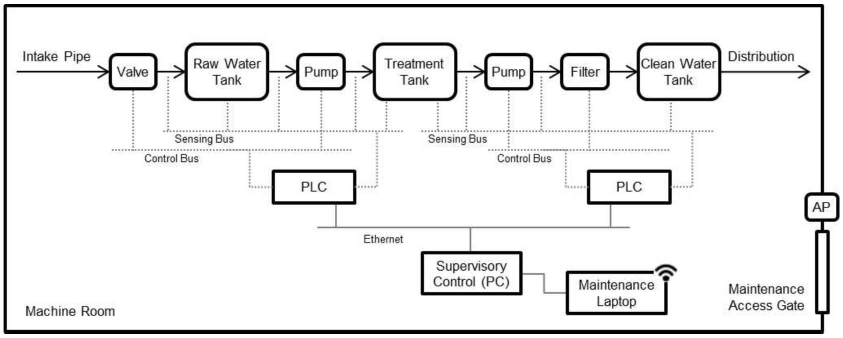

Figure 4. WTP Physical Architecture.

Similarly to [79], inside the WTP there is an initial tank where the raw water is stored. The tank is connected via pipes to several intermediate steps where the water is purified by means of chemicals which are then filtered by filtering procedures (e.g., Uultraviolet filters for dechlorination). The *principal asset* we consider in this example is the physical intergrity of the raw water tank. Two important *indirect assets* are (a) safety of technicians (possibly in the premises of the plant) and (b) building integrity, which may be compromised by the effect of burst and water dispersion. For the sake of simplicity, we suppose there are only the following threats (summarized in Table 1):

1. *Pressure safety control*, the WTP system does not have a safety control against the increase of pressure in the tank above a critical threshold. Therefore, if attacker is able to change the pressure of the raw water tank then he can lead the tank, eventually, to burst. Considering the accessibility of the WTP, its architecture, and the fact that controllers are not connected to the Internet, the only way for the attacker to increase the pressure is by acquiring physical access to the WTP and manipulate the valves and pumps of the system. The *cost for*

the attacker is high since (we supposed) access to the plant requires valid credentials (which the attacker does not know). The *mitigation cost* is high too (8/10) because it requires the installation of an intrusion detection system (plus additional costs to manage the system) and/or the introduction of authentication/encyption schemes in the communication between components. We note that the assumptions on both the vulnerability and the costs are fair since similar situations have been reported on real-world water treatement testbed and plants (see, e.g., [80]).

2. *Ethernet local network*, since the local network is isolated from the Internet, the are no security protocols implemented to guarantee the basic security properties (i.e., confidentiality, integrity, availability). An attacker can easily modify the content of the packets exchanged between PLC. We assume (as in [81]), that the logic of the control of valves and pumps is implemented in the Supervisory Control PC while PLC converts analog to digital messages. Therefore, if the attacker has access to the Ethernet network (similarly to [80]), he can modify the payload of the network packets affecting the control logic of valves and pumps. Considering the accessibility of the WTP, and the isolation from the Internet, the *cost for the attacker* is high. The *mitigation costs* are high too, since the introduction of security mechanisms that guaratee the basic security properties requires the re-ingeneering of the local network. Furthermore, those security mechanisms need to take into account the timing constraints such as the safety response time of the WTP that may result in ad-hoc solutions or delicate fine-tuning of security protocols.

3. *Authentication*, the storage room hosting the WTP can be accessed through authentication on a panel. The panel is subject to (a) physical tampering, (b) attacks through accessible port used for firmware updates, and (c) remote attacks through the access control management server. An attacker that is able to exploit any of these vulnerabilities can gain access to the room. We assume that for an attacker can be realatively easy to bypass authentication procedures since there is a wide number of vulnerabilities and social engineering techniques that have been effectively exploited in the past decades. For the same reason, properly mitigate this risk is not trivial but a number of techniques can be used to mitigate this risk.
4. *USB*, disbling USB ports in a computer connected to a plant increases the security of the overall system with a relative cheap cost of finding other methods to transfer data into the system. On the other hand, if not disabled, many examples (e.g., [17]) showed that USB can be easily exploited by attackers to get access to the system and/or to introduce malwares.

Threats	**Attacker cost**	**Mitigation cost**
Pressure changing	10/10	8/10
Ethernet integrity	9/10	9/10
Authentication bypass	4/10	6/10
USB (SCADA system)	8/10	2/10

5. Table 1 Exploitation and mitigation cost in isolation

An example multi-stage attack that can be identified by the application of a model-based SecRA mixes cyber and physical vulnerabilities with ascending attacker cost.

1. The attacker access to the WTP exploiting social engineering attacks (e.g., piggybacking or tailgating) or integrity vulnerability exploits of the AP.
2. Once the attacker has access to the CPS, he can exploit the USB vulnerability to get access to the network of the plant.
3. Since there are no security mechanisms on the Ethernet network, the attacker can easily alter the payloads of the messages.
4. The attacker alters the nominal behavior of the system (e.g., opening the intake valve and closing the pump after the raw water tank) and burst the water tank.

Such attack shows that the exploitation cost in isolation of Table 1 should change after the discovery of attack chains. In fact, Table 1, the attacker exploitation cost of the USB, and Ethernet threats is high because based on the assumption that access to the CPS is forbidden. However, the attack shows that the attacker cost of exploiting USB, or Ethernet threats is linked to the authentication bypass cost.

Summarizing, instead of estimating the risk as the vulnerability exploitation cost in isolation, model-based analysis techniques can be applied leveraging system architecture and nominal behavior models extended with vulnerabilities. In this way, one could discover complex attack chains and estimate the risks more precisely, based on the interconnection between different vulnerabilities. In our example, the authentication bypass vulnerability has the highest risk because, if not mitigated, can be used to lower the exploitation cost of all the other vulnerabilities.

5 Conclusion

We discussed challenges and opportunities to develop a model-based SecRA for CPS, adopting an asset-driven viewpoint to evaluate risks at system-level, and we identified gaps. We considered modeling languages, techniques, and tools for the SecRA, discussing limitations in their direct application to a CPS. We defined a roadmap of research opportunities for (i) CPS (security) modeling, (ii) vulnerabilities and cost models, (iii) integration with public vulnerability repos, (iii) attacker models and profiles, and (iv) automated discovery of cyber-physical attack chains.

Bibliography

[1] K. Sampigethaya and R. Poovendran, "Aviation cyber-physical systems: Foundations for future aircraft and air transport," *Proceedings of the IEEE,* vol. 101, no. 8, pp. 1834-1855, 2013.

[2] I. Moir, A. Seabridge and M. Jukes, Civil Avionic Systems, John Wiley & Sons, 2013.

[3] M. Shavit, A. Gryc and R. Miucic, "Firmware Update Over The Air (FOTA) for Automotive Industry," in *Asia Pacific Automotive Engineering Conference*, 2007.

[4] M. Howard and S. Lipner, The security development lifecycle, volume 8., Microsoft Press Redmond, 2006.

[5] *ISA/IEC 62443. Security for industrial automation and control systems.*

[6] G. Disterer, "ISO/IEC 27000, 27001 and 27002 for Information Security Management," *Journal of Information Security,* vol. 4, no. 2, pp. 92-100, 2013.

[7] Joint Task Force Transformation Initiative, "SP 800-53 Rev. 4," NIST, 2003.

[8] RTCA Inc, "DO-356," RTCA, 2014.

[9] SAE, "J3061 - Surfacae Vehicle Recommended Practice," SAE International Tech. Rep., 2016.

[10] *ISO/IEC 15408. Information technology – security requirements – evaluation criteria for IT security.*

[11] *The CORAS EU Project FP5 IST-2000-25031,* FP5-IST.

[12] B. S. Blanchard, W. J. Fabrycky and W. J. Fabrycky, Systems engineering and analysis, Prentice Hall Englewood Cliffs, 1990.

[13] J. Rumbaugh, I. Jacobson and G. Booch, Unified Modeling Language Reference Manual (2nd Edition), Pearson Higher Education, 2004.

[14] B. Schneier, "Attack Trees," *Software Tools for the Professional Programmer,* 1999.

[15] A. Shameli-Sendi, R. Aghababaei-Barzegar and M. Cheriet, "Taxonomy of Information Security Risk Assessment (ISRA)," *Journal of Computer Security,* vol. 57, no. C, pp. 14-30, 2016.

[16] J. Shi, J. Wan, H. Yan and H. Suo, "A survey of Cyber-Physical Systems," in *International Conference on Wireless Communications and Signal Processing (WCSP)*, 2011.

[17] S. Weinberger, "Computer security: Is this the start of cyberwarfare?," *Nature News,* vol. 474, no. 7350, pp. 142-145, 2011.

[18] B. Miller and D. Rowe, "A survey SCADA of and critical infrastructure incidents," in *Proceedings of the Conference on Research in information technology*, 2012.

[19] S. Edwards, L. Lavagno, E. Lee and A. Sangiovanni-Vincentelli, "Design of embedded systems: formal models, validation, and synthesis," *Proceedings of the IEEE,* vol. 85, no. 3, pp. 366 - 390, 1997.

[20] A. West, "Nasa study on flight software complexity," NASA, 2009.

[21] OWASP, "The Open Web Applicaiton Security Project (OWASP)," [Online]. Available: www.owasp.org. [Accessed September 2017].

[22] MITRE, "Common Wekness Enumeration (CWE)," [Online]. Available: cwe.mitre.org. [Accessed September 2017].

[23] E. M. Hutchins, M. J. Cloppert and R. M. Amin, "Intelligence-driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains," *Leading Issues in Information Warfare & Security Research,* vol. 1, no. 1, p. 80, 2011.

[24] C. Tankar, "Advanced Persistent threats and how to monitor and deter them," *Network Security,* vol. 2011, no. 8, pp. 16-19, 2011.

[25] ICT, SINTEF, "The CORAS Method," [Online]. Available: http://coras.sourceforge.net/.

[26] "ThreatModeler," [Online]. Available: threatmodeler.com. [Accessed September 2017].

[27] Microsoft Corporation, "STRIDE - Threat Modeling," [Online]. Available: https://msdn.microsoft.com/en-us/library/ff648644.aspx.

[28] RTCA, "DO-333 - Formal Methods Supplement to DO-178C and DO-278A," RTCA, 2011.

[29] B. Blanchet, "Security protocol verification: symbolic and computational models," in *International Conference on Principles of Security and Trust (POST)*, 2012.

[30] A. Refsdal, B. Solhaug and K. Stolen, "Cyber Risk Management," in *Cyber Risk Management*, Springer, 2015, pp. 33-47.

[31] International Organization for Standardization, *ISO 31000 - Risk management - Principles and guidelines,* 2009.

[32] 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, no. C, pp. 1 - 27, 2016.

[33] NIST, "Cybersecurity Framework," [Online]. Available: https://www.nist.gov/cyberframework.

[34] A. Shostack, Threat Modeling: Designing for Security, Wiley, 2014.

[35] "SESAR," [Online]. Available: www.sesarju.eu.

[36] M. S. Lund, B. Solhaug and K. Stølen, The CORAS Approach, Springer, 2011.

[37] NIST, "National Vulnerability Database (NVD)," [Online]. Available: nvd.nist.gov . [Accessed September 2017].

[38] OMG, "UML succsess stories," [Online]. Available: http://www.uml.org/uml_success_stories/index.htm. [Accessed September 2017].

[39] S. H. Houmb, F. Den Braber, M. S. Lund and K. Stølen, "Towards a UML profile for model-based risk assessment," in *Workshop on Critical systems development with UML*, 2002.

[40] M. S. Lund, I. Hogganvik, F. Seehusen and K. Stølen, "UML Profile for Security Assessment," Techinical report STF A, 2003.

[41] T. Lodderstedt, D. Basin and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *Proceedings of the International Conference on the Unified Modeling Language*, 2002.

[42] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *Proceedings of the International Conference on the Unified Modeling Language*, 2002.

[43] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proceedings of Computer Security Applications Conference (ACSAC)*, 1999.

[44] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements engineering,* vol. 10, no. 1, pp. 34 - 44, 2005.

[45] T. Weilkiens, Systems Engineering with SysML/UML: Modeling, Analysis, Design, The OMG Press, 2007.

[46] Y. Roudier and L. Apvrille, "SysML-Sec: A Model Driven Approach for Designing Safe and Secure Systems," in *Model-Driven Engineering and Software Development Conference (MODELSWARD)*, 2015.

[47] F. Lugou, L. W. Li, L. Apvrille and R. Ameur-Boulifa, "Sysml models and model transformation for security," in *Model-Driven Engineering and Software Development Conference (Modelsward)*, 2016.

[48] *E-safety Vehicle Intrusion Protected Applications (EVITA) EU FP7 Programme,* 2007-2013.

[49] AADL. [Online]. Available: http://www.aadl.info/. [Accessed March 2018].

[50] R. Ellison, A. Householder, J. Hudak, R. Kazman and C. Woody, "Extending AADL for Security Design Assurance of Cyber-Physical Systems," CMU/SEI-2015-TR-014.

[51] M. Rocchetto and N. O. Tippenhauer, "Towards formal security analysis of industrial control systems," in *Asia Conference on Computer and Communications Security (AsiaCCS)*, 2017.

[52] C. M. Ahmed, C. Murgia and J. Ruths, "Model-based Attack Detection Scheme for Smart Water Distribution Networks," in *Asia Conference on Computer and Communication Security (AsiaCCS)*, 2017.

[53] M. Rocchetto and N. O. Tippenhauer, "On attacker models and profiles for cyber-physical systems," in *European Symposium on Research in Computer Science (ESORICS)*, 2016.

[54] R. Lanotte, M. Merro, R. Muradore and L. Viganò, "A formal approach to cyber-physical attacks," in *Computer Security Foundation Symposium (CSF)*, 2017.

[55] C. Herley, "Unfalsifiability of security claims," *National Academy of Sciences,* vol. 113, no. 23, pp. 6415-6420, 2016.

[56] B. Blanchet, "Modeling and verifying security protocols with the applied pi calculus and ProVerif," *Foundations and Trends in Privacy and Security,* vol. 1, no. 1-2, pp. 1-135, 2016.

[57] D. Garlan, "Formal modeling and analysis of software architecture: Components, connectors, and events," in *Proceedings of Formal Methods for Software Architectures*, 2003.

[58] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science,* vol. 324, no. 5923, pp. 81-85, 2009.

[59] S. Schupp, E. Abraham, X. Chen, I. B. Makhlouf, G. Frehse, S. Sankaranarayanan and S. Kowalewski, "Current Challenges in the Verification of Hybrid Systems," in *CyPhy 2015, LNCS 9361, pp. 8–24,*, 2015.

[60] A. Platzer, Logical Analysis of Hybrid Systems, Springer, 2010.

[61] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems Conference (TACAS)*, 2008.

[62] D. Jovanović and L. de Moura, "Solving non-linear arithmetic," in *International Joint Conference of Automated Reasoning (IJCAR)*, 2012.

[63] B. Dutertre, "Yices 2.2," in *Computer Aided Verification (CAV)*, 2014.

[64] A. Cimatti, A. Griggio, B. J. Schaafsma and R. Sebastiani, "The MathSAT5 SMT Solver," in *Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2013.

[65] C. Barrett, C. L. Conway, D. Morgan, L. Hadarean, D. Jovanović, T. King, A.

Reynolds and C. Tinelli, "Cvc4," in *International Conference on Computer Aided Verification (CAV)*, 2011.

[66] A. Cimatti, A. Griggio, A. Irfan, M. Roveri and R. Sebastiani, "Invariant Checking of NRA Transition Systems via Incremental Reduction to LRA with EUF," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2017.

[67] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transacytions on Information Theory,* vol. 29, no. 2, pp. 198-208, 1983.

[68] S. Escobar, C. Meadows and J. Meseguer, "A rewriting-based inference system for the nrl protocol analyzer and its meta-logical properties," *Theoretical Computer Science,* vol. 367, no. 1-2, pp. 162-202, 2006.

[69] D. Basin, S. Capkun, P. Schaller and B. Schmidt, "Let's Get Physical: Models and Methods for Real-World Security Protocols," in *International Conference on Theorem Proving in Higher Order Logics (TPHOL)*, 2009.

[70] M. S. Barik, A. Segupta and C. Mazumdar, "Attack graph generation and analysis technique," *Defence Science Journal,* vol. 66, no. 6, pp. 559-567, 2016.

[71] J. A. Wang and M. Guo, "Ovm: An ontology for vulnerability management," in *Workshop on Cyber Security and Information Intelligence Research (CSIIRW)*, 2009.

[72] M. Felderer, P. Zech, R. Breu, M. Büchler and A. Pretschner, "Model-based security testing: a taxonomy and systematic classification," *Software Testing, Verification and Reliability,* vol. 26, no. 2, pp. 119-148, 2016.

[73] P. Mell, K. Scarfone and S. Romanosky, "Common vulnerability scoring system," *IEEE Security & Privacy,* vol. 4, no. 6, 2006.

[74] P. Mell and T. Grance, "Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme," National Insitute of Standards and Technology Gaithersburg MD Computer Security Division, 2002.

[75] MITRE, "Open Vulnerbility and Assessment Language (OVAL)," [Online]. Available: https://oval.cisecurity.org/. [Accessed September 2017].

[76] MITRE, "Common Attack Pattern and Enumeration and Classification (CAPEC)," [Online]. Available: http://capec.mitre.org/. [Accessed September 2017].

[77] M. Glinz, "An integrated formal model of scenarios based on statecharts," in *Software Engineering (ESEC)*, 1995.

[78] A. Arnold, M. Baleani, A. Ferrari, M. Marazza, V. Senni, A. Legay, J. Quilbeuf and C. Etzien, "An Application of SMC to continuous validation of heterogeneous systems," in *SimuTools*, 2016.

[79] A. P. Mathur and N. O. Tippenhauer, "SWaT: A water treatment testbed for research and training on ICS security," in *Proceedings of the Cyber-physical Systems for Smart Water Networks (CySWater) workshop*, 2016.

[80] D. Urbina, J. Giraldo, N. O. Tippenhauer and A. Cardenas, "Attacking Fieldbus Communications in ICS: Applications to the SWaT Testbed," in *Proceedings of Singapore Cyber Security Conference (SG-CRC)*, 2016.

[81] M. Rocchetto and N. O. Tippenhauer, "CPDY: Extending the Dolev-Yao Attacker with," in *International Conference on Formal Engineering Methods (ICFEM)*, 2016.

[82] M. Gario and A. Micheli, "Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms," in *International Workshop on Satisfiability Modulo Theories (SMT)*, 2015.

[83] MITRE, "CVEDetails - Top 50 Products by Total Number of Distinct Vulnerabilities in 2017," [Online]. Available: https://www.cvedetails.com/top-50-products.php?year=2017. [Accessed September 2017].

[84] K. D. Evensen and K. Weiss, "A comparison and evaluation of real-time software systems modeling languages," in *Aerospace Conference*, 2010.

[85] Q. He, X. Meng and R. Qu, "Survey on cyber security of CAV," in *Forum on Cooperative Positioning and Service (CPGPS)*, 2017.

[86] F. Marcus, P. Wolfgang and R. Bernhard, The UML profile for framework architectures, Addison-Wesley Longman Publishing Co., Inc., 2000.

[87] "Microsoft threat modeling tool," [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=49168.

[88] R. Sebastiani and P. Trentin, "OptiMathSAT: a tool for optimization modulo theories," in *International Conference on Computer Aided Verification (CAV)*, 2015.

[89] N. Bjørner, A. D. Phan and L. Fleckenstein, "νZ-An Optimizing SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems Conference (TACAS)*, 2015.

[90] A. Calvi and L. Viganò, "An Automated Approach for Testing the Security of Web Applications Against Chained Attacks," in *Symposium on Applied Computing (SAC)*, 2016.

[91] M. Urbanska, I. Ray, A. E. Howe and M. Roberts, "Structuring a Vulnerability Description for Comprehensive Single System Security Analysis," in *Rocky Mountain Celebration of Women in Computing (RMCWiC)*, 2012.

[92] M. Yampolskiy, P. Horvath, X. D. Koutsoukos, Y. Xue and J. Sztipanovits, "Taxonomy of Description of Cross-domain Attacks on CPS," in *High Confidence Networked Systems Conference (HiCoNS)*, 2013.

[93] MITRE, "Risk Management Tools," [Online]. Available: https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/risk-management/risk-management-tools. [Accessed September 2017].