Hacks *et al. Energy Informatics* (2020) 3:30
https://doi.org/10.1186/s42162-020-00134-4

Energy Informatics

SOFTWARE Open Access

powerLang: a probabilistic attack simulation language for the power domain

Simon Hacks*, Sotirios Katsikeas, Engla Ling, Robert Lagerström and Mathias Ekstedt

*Correspondence: shacks@kth.se
Division of Network and Systems Engineering, KTH Royal Institute of Technology, Teknikringen 33, 114 28 Stockholm, Sweden

Abstract

Cyber-attacks on power-related IT and OT infrastructures can have disastrous consequences for individuals, regions, as well as whole nations. In order to respond to these threats, the cyber security assessment of IT and OT infrastructures can foster a higher degree of safety and resilience against cyber-attacks. Therefore, the use of attack simulations based on system architecture models is proposed. To reduce the effort of creating new attack graphs for each system under assessment, domain-specific languages (DSLs) can be employed. DSLs codify the common attack logics of the considered domain.

Previously, MAL (the Meta Attack Language) was proposed, which serves as a framework to develop DSLs and generate attack graphs for modeled infrastructures. In this article, powerLang as a MAL-based DSL for modeling IT and OT infrastructures in the power domain is proposed. Further, it allows analyzing weaknesses related to known attacks. To comprise powerLang, two existing MAL-based DSL are combined with a new language focusing on industrial control systems (ICS). Finally, this first version of the language was validated against a known cyber-attack.

Keywords: Threat modeling, Attack simulation, Domain specific language, Power domain, Industrial control systems

Introduction

Recent deliberate disruptions of electrical power and energy systems (Defense Use Case 2016; Petermann et al. 2011) have shown that cyber-attacks on power assets can have disastrous consequences for individuals, regions, and whole nations. Attackers exploit malicious code to manipulate the controls of power grids, energy providers, and other critical infrastructure (Liu et al. 2011; Wang and Rong 2009). These manipulations result in real-world catastrophic physical damage, like major power outage or city-wide disruptions of any service that requires electric power (Defense Use Case 2016; Petermann et al. 2011; Rosas-Casals et al. 2007). In order to respond to these threats, the assessment of power domain's cyber security fosters a higher degree of safety for the entire society dependent on electric power.

Unfortunately, assessing the cyber security of an entire domain and its single entities is difficult. To identify vulnerabilities, security-relevant parts of the system must be understood, and all potential attacks need to be identified (Morikawa and Yamaoka 2011).

There are three challenges that need to be solved: First, all relevant security properties of a system need to be identified. Next, it is difficult to collect necessary information on the systems. Last, the collected information needs to be processed to detect all weaknesses that can be exploited. To address this, the use of attack simulations based on system architecture models have been proposed (e.g., Ekstedt et al. 2015; Holm et al. 2015). These approaches take a model of the system and simulate cyber-attacks to identify weaknesses. In other words, this is a execution of many parallel virtual penetration tests. This allows the security assessor to focus on the collection of the information about the system that is necessary for the simulations.

As the previous approaches rely on a static implementation, the use of MAL (the Meta Attack Language) (Johnson et al. 2018) was proposed. This framework for domain-specific languages (DSLs) defines which information about a system is required and specifies the generic attack logic. Since MAL is a meta language (i.e., the set of rules that should be used to create a new DSL), it does not define a particular domain of interest. Therefore, this work aims to create and evaluate a MAL-based DSL for simulation of known cyber-attacks for the power domain.

To achieve this goal, we reuse two existing MAL-based DSL: coreLang (Katsikeas et al. 2020), which is designed to model common IT infrastructures, and sclLang (Ling 2020), which covers the internal structure of substations. To bridge the gap between the IT world, presented by coreLang, and the technical world in sclLang, we propose icsLang to provide a means for modelling OT environments.

The rest of this work is structured as follows: First, we present the state of the art; second, we explain the objectives towards our language; then, we provide the needed knowledge to create powerLang and the way it is comprised; next, demonstrate powerLang based on the Ukrainian scenario and discuss the generated insights, before we conclude our work.

State of the art

Our work relates to two domains of previous work: model-driven security engineering, attack/defense graphs, and security assessment in the power domain.

Within the domain of model-driven security engineering a large number of domain-specific languages has been proposed (Alam et al. 2007; Basin et al. 2011; Jürjens 2005; Paja et al. 2015). These languages typically allow to model a system's design with respect to its components and the interaction among these. Additionally, these languages encompass security properties such as constraints, requirements, or threats. Different formalisms and logics are used to provide the opportunity for model checking and searching for constraint violations. However, not all languages support automated analysis purposes (Almorsy and Grundy 2014; Lund et al. 2010). Instead, they solely offer the capability to model security relevant properties and analysis needs to be conducted manually.

The concept of attack trees was made popular by the work of Bruce Schneier (1999; 2000), further formalized by Mauw and Oostdijk (2005), and extended to include defenses by Kordy et al. (2010). There are several approaches elaborating on attack graphs (Ingols et al. 2009; Kordy et al. 2014; Williams et al. 2008). These theoretical descriptions led to the development of different tools using attack graphs, which mostly build up on collecting information about existing systems and automatically create attack graphs. For

example, the TVA tool (Noel et al. 2009) models security conditions in networks and enriches these by exploits that describe the transitions between these security conditions.

These attack graphs can be extended to probabilistic attack graphs, e.g., by facilitating Bayesian networks. Frigault et al. (2008) use the TVA-tool to generate attack graphs, ad transform them to dynamic Bayesian networks. Finally, they enrich them with probabilities using CVSS (Common Vulnerability Scoring System) scores, like (Xie et al. 2010), who model uncertainties in the attack structure, attacker's actions, and alerts triggering.

Hitherto, we have united the approaches of attack graphs and system modeling in our previous works like P2CySeMoL (Holm et al. 2015) and securiCAD (Ekstedt et al. 2015). Our central idea was to automatically generate probabilistic attack graphs based on a existing system specification. The generated attack graph then serves as an inference engine and produces predictive security analysis results from the system model. However, in these works, the used languages to create the attack graphs were hard-coded. Therefore, we have proposed MAL that allows to create domain specific languages. So far, several languages have been built in MAL. One example is vehicleLang (Katsikeas et al. 2019), which allows modeling cyber-attacks on modern vehicles. Another example is coreLang, which contains the most common IT entities and attack steps and is included in the presentation of MAL (Johnson et al. 2018). Further, we proposed the automated creation of MAL languages by translating existing concepts from the ArchiMate language to MAL (Hacks et al. 2019).

Next, we discuss works, which elaborate on structural and security related aspects in the power domain. First, (Jiang et al. 2018) propose a DSL and a repository to represent power grids and related IT components. The SGAM (Smart Grid Architecture Model) (CEN-CENELEC-ETSI 2012) describes a technical reference architecture that describes the functional information flows between the main domains of smart grids. Further, it integrates several systems and subsystems architectures.

Cherdantseva et al. (2016) conducted a systematic literature review to identify different methods to assess risk in Supervisory Control and Data Acquisition (SCADA) systems. Another review was conducted by Franke and Brynielsson (2014) considering cyber situational awareness. They cluster their findings, relate it to national cyber strategies, and give suggestions for future research. Sharifi and Yamagata (2016) research risks for the resilience of urban power supply and identify cyber threats as one origin of risk.

Additionally to the reviews on existing literature, researchers have conducted other kinds of surveys as well. Wang and Lu (2013) assessed different challenges that arise from cyber threats for the smart grid domain. Masood (2016) conducts similar research for cyber threats on nuclear power plants and, additionally, provides attack trees to model the threats.

Finally, we discuss approaches that help to improve the security in the power domain. Habash et al. (2013) provide a framework to overcome the cyber-physical threats of the power grid and have evaluated it in Canada by applying it for one year. Ten et al. (2008) suggest a vulnerability assessment to systematically evaluate the vulnerabilities of SCADA systems at the three levels of system, scenarios, and access points.

Objectives

Our creation of powerLang follows a design-centered approach, as we have the existing means of MAL and design a language that enables stakeholders in the power domain to

easily assess threats to their IT and OT environments. This is needed as the responsibles in the power domain usually prioritize availability and resilience of the power supply (Wang and Lu 2013) and, thus, lack knowledge on security related aspects.

The entire language development is embedded in a bigger project that will equip utility providers with a fully-fletched framework of integrated tools that will improve the security in the power grid (EnergyShield 2020).

The value chain in the power domain is comprised of five main categories of stakeholders: bulk generation, transmission, distribution, distributed energy resource, and customer premises (CEN-CENELEC-ETSI 2012). To create a language covering the entire chain is ambitious. Therefore, we opt for an iteratively, continuously extension of our language to cover further aspects in the domain. In our first iteration, we opted to represent the domain of transmission, as this was prioritized in our project, due to reachability of the practitioners.

The main concern of our practitioners was the ability to have existing attacks, like the Ukraine scenario (Defense Use Case 2016), included in with the language created models. Thus, the language needs to cover IT as well as OT parts. Additionally, we received a list of concrete assets that they use in their OT environment. Thus, the objectives towards our language can be characterized as follows:

- Known attacks should be discovered by performing simulations, which use powerLang.
- The language should reflect the different needs of IT and OT environments.
- The terminology of the power domain should be reflected in the language.

Creating powerLang

In this section, we describe how powerLang is comprised. Therefore, we take a closer look at the reused languages, icsLang that bridges the identified gap between these two languages, and the mechanics that we apply to combine these languages.

MAL and reused languages

Before we present powerLang as a whole, we give first a short introduction to MAL so that the reader can understand the mechanics of MAL-based languages. Further, we give an overview of the reused languages. For detailed insights, we refer to the original publications introducing MAL (Johnson et al. 2018), coreLang (Katsikeas et al. 2020), and sclLang (Ling 2020).

Introduction to MAL

Hitherto, we proposed the use of system architecture models to conduct attack simulations (e.g., Ekstedt et al. 2015; Holm et al. 2015). However, our previous approaches used static implementations to represent assets and the related security information within the tools. Thus, changes to these underlying structures require high effort. Therefore, we developed MAL (Johnson et al. 2018). MAL itself defines which information about a system is required and specifies the generic attack logic. MAL is a meta language (i.e. the set of rules that should be used to create a new DSL) and represents no particular domain of interest. But it is to be used for creating languages that represent a particular domain. Before we explain such a concrete language, we present the basic building blocks of MAL, to ease the understanding of these languages.

First, a MAL language contains so called *assets*, which are the main elements found in the domain under study. Next, assets contain *attack steps*, which are actual attacks/threats that can be exploited. An attack step can be connected with one or more following attack steps to create an attack path. The sum of the attack paths are attack graphs used for the simulation. Assets can be linked by *associations* to model possible transitions of an attacker among them. Further, inheritance between assets is allowed as known from object orientation. Finally, there exists categories that allow organization of assets and probability distributions can be assigned to attack steps. A probability expresses the effort needed to complete the related attack step.

Next, we give a short example how a MAL language looks like. This example, which is a snippet of a complete language, contains attack steps on three assets. It illustrates how the attack steps are connected with each other, for example if one achieves *blockingOperation*, one is able to reach *overspeed* on `Turbine` and as a result finally lead to *plantDamage* and *powerOutage* on the power plant. Additionally, *blockingOperation* is annotated with *Normal(5,0.2)* meaning that an attacker's effort is described by a normal distribution with a mean of 5 and a standard deviation of 0.2. In the last lines of the example, the associations between the assets are defined.

```
category PowerPlantAssets {
  ...
  asset PowerPlant extends Facility
  {
    | plantDamage
      -> powerOutage
    | powerOutage
      -> city.blackout
  }
  asset Turbine extends RotatingEquipment
  {
    | systemFailure
      -> plant.powerOutage
    | overspeed
      -> systemFailure,
         plant.plantDamage
  }
  asset Controller extends Equipment
  {
    | closeValves
      -> controlledSystem.shutDown
    | reduceFlow
      -> controlledSystem.materialFailure
    | blockingOperation [Normal(5,0.2)]
      -> rotatingEquipment.overspeed
    | influenceMeasurement
      -> controlledSystem.manipulate
    }
  ...
}
associations {
  PowerPlant  [plant] 1 <-- ComprisedOf --> *
      [controllers]   Controller
  PowerPlant  [plant] 1 <-- ComprisedOf --> *
```

```
      [equipment] RotatingEquipment
  Controller    [controller] 1 <-- Controls --> 1
      [equipment] RotatingEquipment
  Controller    [controller] 1 <-- Controls --> *
      [system]  ControlledSystem
}
```

coreLang

As illustrated before, MAL provides the basics to create a threat modeling language from scratch. However, many languages created with MAL share a common set of concepts. To reduce unnecessary redundant work, we developed coreLang (Katsikeas et al. 2020) that is comprised of predefined assets that appeared in different languages created with MAL. Thus, this coreLang can serve as starting point to model more domain specific languages or even act as a rudimentary language to model simple environments. In the following, we will give a short overview on the basic elements of this language and refer to the original publication for all the details.

Figure 1 presents the overall structure of coreLang. The extends relationship expresses an inheritance between the parent and the child assets. Consequently, all attack steps and defenses of the parent asset are available in the child, too. However, it is possible to either complement the existing attack steps with further logic or to completely overwrite the logic. In contrast, the association relationship expresses possible paths for attackers between different assets. In other words, the associations describe the links between assets that can be used by attackers to traverse from one asset to another. For coreLang, we have identified six different main categories: system, vulnerability, user, IAM (Identity and Access Management), data resources, and networking. In the following, we will describe those categories and the related design decisions.

System The first category, we like to shed light on, is *system*. This is the collection of assets that usually represent the computing instances in an environment, and thus are the

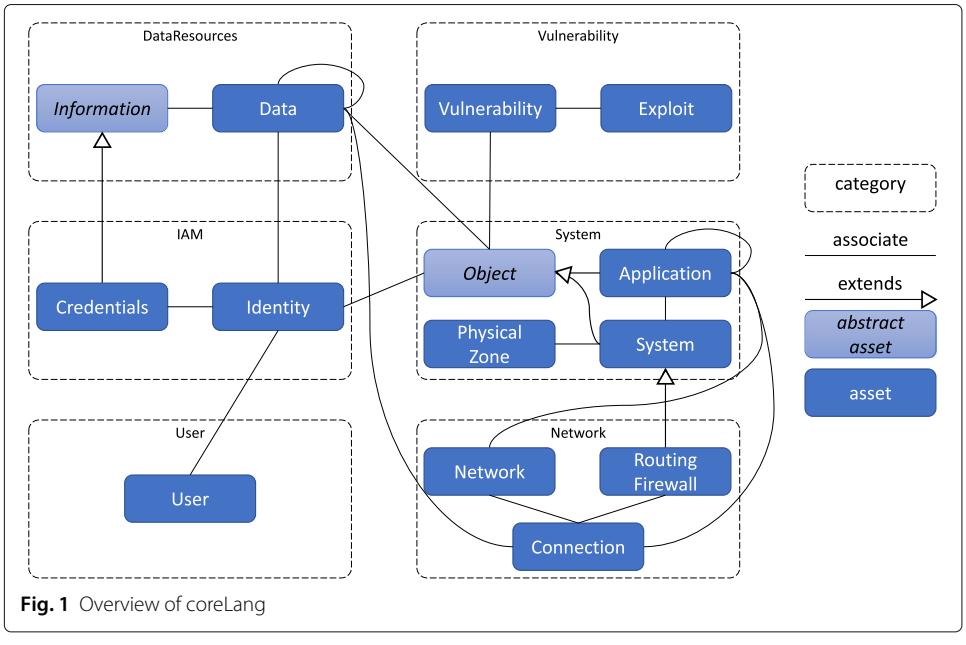

Fig. 1 Overview of coreLang

natural attack surface. First, we created an asset called `Object` (inspired by the object in object-oriented programming languages) that provides common functionality to all inheriting assets. Basically, an `Object` is the simplest form of an asset that can be compromised by a `Vulnerability`. On the one hand, `Object` is specialized into `System`, which specifies the hardware on which `Applications` can run. The attacker can DoS everything that is running on it and access (using physical control) on the OS after effort. On the other hand, `Object` is specialized into `Application`, which specifies pretty much everything that is executed or can execute other applications. Lastly, this category contains `PhysicalZone`, which is the location where systems are physically deployed.

Vulnerability The basic idea of creating a MAL related language is to provide a set of already known attack steps to the modeller. However, this incorporates two types of shortcomings. First, we concentrate on known attack steps. But, there are also attack steps that are not known yet. Second, we stay on a relatively abstract level for coreLang. Consequently, we cannot provide all possible attack steps upfront, as the attack steps are very diverse for different assets. To overcome this issue, we provide a set of `Vulnerability` and `Exploit`. On the one hand, these assets can be used as extension points for other language developers. On the other hand, we provide a standard set of `Vulnerability` and `Exploit`. These can be used by the end-user to model attack steps that are not known at the time of creating the language. Basically, an `Object` does have a `Vulnerability` that can have different levels of complexity to take advantage of. This `Vulnerability` can then be facilitated by an `Exploit` that leads to different levels of access to the exploited `Object`.

User This category contains only the representation of a `User`. The `User` serves as attack surface for social engineering attacks. The most apparent attack modeled in this asset is the *phishing*, which can lead to either *credential theft* or *takeover* of the user's computer. However, this asset will be extended to represent more attacks in future iterations.

IAM is an accepted way to manage different identities representing users and their access to certain applications (Witty et al. 2003). Therefore, the category IAM is comprised of `Identity` that visualizes a user group. After authentication or compromise of an `Identity`, the attacker assumes its privileges. This is leads to both legitimate and illegitimate access. This access to an `Identity` is usually secured by means of `Credentials`. Those `Credentials` can be stolen/guessed by the attacker directly (e.g., due to brute-force) or the `User` can be convinced to enter them by herself (e.g., due to social engineering).

DataResources This category groups the assets that are usually communicated. First, there is `Data` that represents any form of data that can be stored or transmitted. An attacker can perform the classical actions of read, write, and delete. Second, we have defined `Information` as an abstract concept that is incorporated in `Data`.

Networking The last category elaborates on networking related assets. First, we have identified the `Network` where a network zone is a set of network accessible applications. The border of such a `Network` is a `RoutingFirewall` that specifies a router with firewall capabilities that connects many networks. Lastly, there are `Connections` between `Applications` that allow a communication along different `Networks`, and consequently, a lateral movement of an attacker.

sclLang

The standard IEC 61850 was developed as substations became increasingly automated and digitalized to help with the transition (IEC Standard 2003). One part of this standard is the Substation Configuration Description Language (SCL). SCL was created to help with compatibility of different vendors and to share configurations of Intelligent Electronic Devices (IEDs) (IEC 2018). Because modern substations built according to the IEC 61850 standard already have existing configuration files according to SCL, we decided to build a DSL based on SCL so that this already existing information could be used to create threat models.

In sclLang (Ling 2020), the assets and their associations are precisely as specified in SCL as seen in Fig. 2. These assets are divided into three different categories. The *Functional* assets are related to the main substation functionality and includes, for example, transformers that are most often used to alter the voltage level. The *Product* category includes the products used in a substation, for example the IEDs that are used to enable automation. Finally, the *Communication* category includes all assets that are needed for the communication of the IEDs.

Regarding attacks, the attack steps in sclLang are *access*, *communicate*, *execution*, *impact* and *hasRouter*.

The attack steps *access*, *execution* and *impact* are from the categories of tactics as found in the ATT&CK Matrix for Enterprise (MITRE 2020a). Please note that *access* has been renamed from the category Initial Access. The categories of attacks were used to keep the complexity of the attacks down in the first version of the language. The attack step

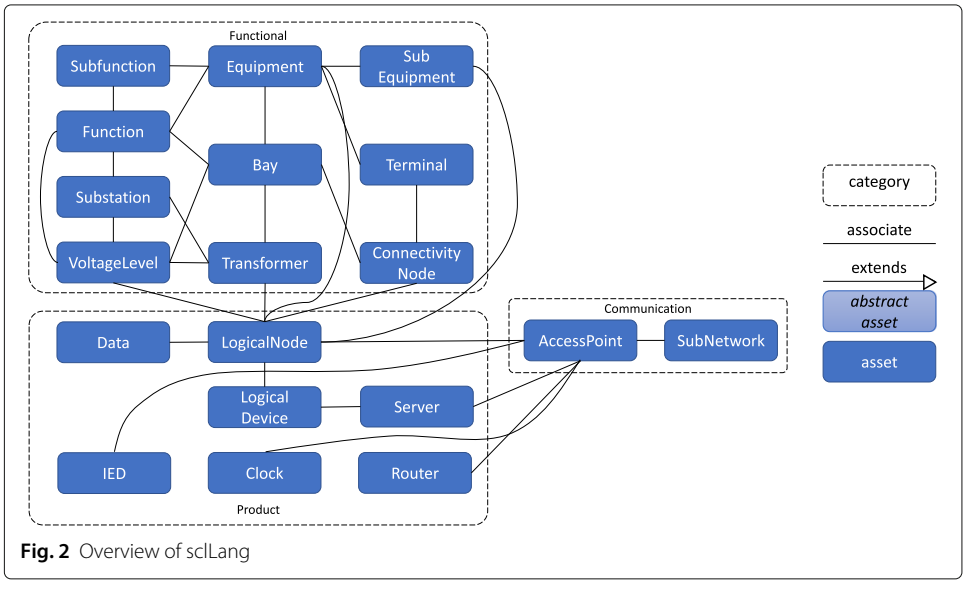

Fig. 2 Overview of sclLang

communicate was added to model how an attacker may communicate throughout the substation. Finally, the attack step *hasRouter* was added because whether an IED has the routing function enabled or not affects the attacker possibility to move throughout the substation.

Access The attack step *access* is the first attack that an external attacker would perform to gain initial access to a substation. It is possible that, for instance, a disgruntled employee would not need this attack step. After the attacker has access, they can move further through the substation by reaching other attack steps.

Communicate The modern IEC 61850 substations are built with communication networks and this attack steps shows how the attacker would be able to move through the network and therefore also the substation. Most of the communication happens through the access points, which can be physical or logical interfaces. If one IED has two different access points that are part of two different subnetworks, it is possible for the attacker to move between the subnetworks in a substation.

Execution The term execution in attacks is often defined as running malicious code. In a substation it is not necessarily code that creates certain actions but instead sending and receiving logical nodes. Logical nodes are sent for automation purposes to increase the voltage when a specific voltage level is detected. In this sense, one can consider that sending logical nodes is similar to execution of malicious tasks in an IT system. If an attacker reaches the *execution* attack step it is possible to potentially shut down the entire substation.

Impact The attack step *impact* means that the attacker has managed to in some way made an alteration in the substation. As described above in *execution*, the substation automation happens with logical nodes. This means that if an attacker can make an impact on a logical node, perhaps make the voltage level seem lower than it actually is, a malicious execution to increase the voltage level can occur and cause damage to the substation. It is also possible for the attacker to use that *impact* attack step to alter the clock, which is essential to the synchronization of substations and cause disruptions in this way.

HasRouter The attack step *hasRouter* is reached only if the router functionality is enabled on an `IED`. Similar to IT networks a router is needed to move between subnetworks in a substation. With this attack step it is possible for an attacker to move throughout the substation in between subnetworks.

In terms of MAL, this means that the `AccessPoint` can either have a `Router` existing or not. If the router exists, then the attacker can reach the attack step and communicate across `Subnetworks`. However, if no `Router` exists the attacker will not succeed to communicate.

icsLang

Following, we will focus on icsLang (depicted in Fig. 3) that is based on the ATT&CK Matrix for ICS (MITRE 2020b). The main asset in this language is represented by

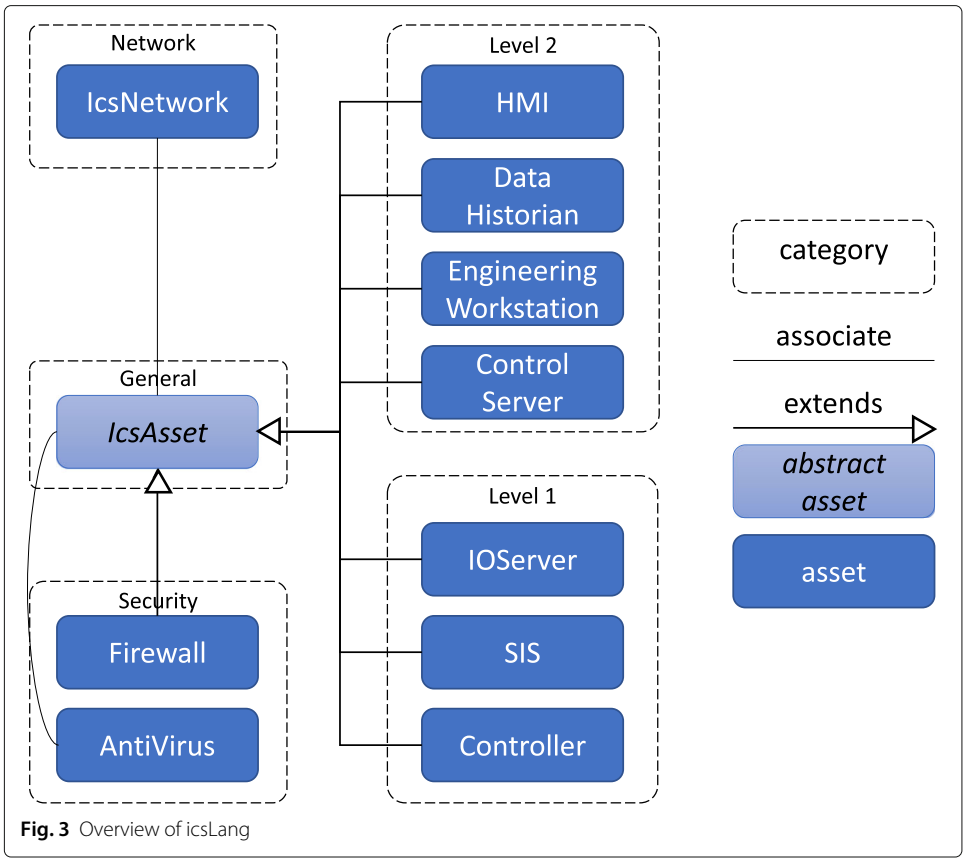

Fig. 3 Overview of icsLang

the `IcsAsset`. It represents the abstract common behavior of all represented assets. Therefore, we have created a connection to `IcsNetwork` to illustrate that assets can communicate with each other if they are attached to the same network. The rest of the language is structured along the MITRE ATT&CK categories level 2 (supervisory control), level 1 (control network), and security.

Level 2 The supervisory control LAN level includes the functions involved in monitoring and controlling physical processes and the general deployment of systems. The central asset on this layer is the `ControlServer` which operates the `Controller` on level 1 and also computes their output (Stouffer et al. 2011). Those `ControlServer` and `Controller` are configured, maintained, and diagnosed using `EngineeringWorkstations`. To carry on the entire system, the operators use human machine interfaces (`HMI`) that provide a graphical user interface to `EngineeringWorkstations` and `ControlServer`. Lastly, there is a `DataHistorian` on level 2, which provides access to external users that are interested in data access for archival or analysis.

Level 1 The control network level includes the functions involved in sensing and manipulating physical processes. This is usually done by `Controller` that are controlled by `ControlServer` situated on level 2. However, this connection is not direct, but through an `IOServer` that provides the interface between the control system LAN applications and the field equipment monitored and controlled by the control system applications.

Additionally, there exists a safety layer consisting of safety instrumented systems (`SIS`). The function of protective relaying is to cause the prompt removal from service of an element of a power system when it suffers a short circuit or when it starts to operate in any abnormal manner that might cause damage or otherwise interfere with the effective operation of the rest of the system.

Security Originally, the ATT&CK Matrix for ICS does not include security related assets explicitly. However, for every asset, there are mitigation defined that can be also modelled as assets. First, we introduced an `AntiVirus` that can detect malicious files on connected assets leading to a higher effort an attacker has to spend. Second, we added a `Firewall` that can block certain ports, enforce white lists, or apply intrusion detection systems (IDS).

Attack Steps The ATT&CK Matrix for ICS contains 81 different techniques that can be used to exploit an ICS environment. Every of these techniques can be an attack step in our icsLang. As the justification if a certain technique will be included into the language would be quite extensive, we will focus on the overarching eleven tactics similar to the approach followed in substation-DSL. Based on the techniques, we will argue if a related technique will be included as attack step or not.

When we create MAL languages, we assume worst case scenarios, meaning that the attacker already knows about the architecture of the attacked organization. This leads to the decision that techniques related to collection, discovery, and lateral movement does not needed to be modelled in icsLang. Additionally, command and control related techniques that describe the way the attacker connects to the environment are not taken into account.

Another set of techniques, we will not model explicitly in our language, are related to the techniques of evasion, persistence, and inhibit response function. These techniques aim to masquerade the doing of the attacker or to create hooks which should prevent the lock out of the attacker by countermeasures. This cannot be represented in MAL directly but is already included in the given probabilities that a certain attack step is successful.

The rest of the techniques follow the common attacker's behavior pattern of access, privilege escalation, and harm the system (Ramsbrock et al. 2007). First, the techniques related to initial access provide either the initial attack surface or the entry-point to an asset due to lateral movement through the network. If the access is established, the attacker can perform techniques related to execution to perform a privilege escalation. Reaching this stage allows to perform attack steps that end up in impact on the asset or impair process control. Impact means in this case that ICS systems, data, and their surrounding environment can be manipulated, interrupted, or destroyed. Impair process control leads to manipulation, deactivation, or physical damage on control processes.

Probability Distributions on Attack Steps For MAL based languages it is essential to define probability distribution on attack steps and defenses to describe how much effort an attacker has to spend to exploit certain attack steps or to which degree a defense is successful. Unfortunately, research that assess such probability distributions is quite scarce and often we have to rely on expert knowledge to model them. In the following, we will

give an overview of the applied probability distributions that can be found in literature. Nonetheless, we like to highlight that the following distributions reflect not the entire reality but rather are approximately reasonable as long as no more solid studies on each and every attack step has been conducted. Additionally, the main contribution of this paper are the structures provided before and not the figures presented next.

First, (Baggett 2008) has identified that if anti-malware is enabled on a system, an attacker's success probability decreases to 90%. Therefore, we model a defense on different attack steps such as *modifyControlLogic* or *rootkit* with a Bernoulli distribution of 0.9. This means that if an anti-malware is in place it will be effective in 10% of the cases.

Second, research (Holm 2014; Sommestad and Hunstad 2013) has elaborated on identifying the effort that an attacker needs to spend to bypass an IDS. This effort depends on different parameters as the IDS is tuned and updated regularly. However, due modeling reasons we assume the worst case that the IDS is neither patched nor updated leading to an exponential distribution with a median of 3.5 days.

Third, from an interview with a domain expert in combination with vulnerability data from the US National Vulnerability Database (NVD), we conclude that a man in the middle attack is successful in 99% if no defense is in place and in 1% if the communication is encrypted. Accordingly, we model defenses with corresponding Bernoulli distributions.

Fourth, the probabilities of finding an entrance in a misconfigured firewall are gathered from a non-published expert survey performed in June 2016. According to the survey, a one day effort will result in a 28% probability of finding an entrance, after ten days the probability is 55%. The estimated gamma parameters then become 0.33, and 74.

Linking languages

So far, we have presented different languages that can be used to model different parts in the energy domain. However, to provide a fully functional language that covers the needs of the energy domain, these languages need to be linked to each other. Generally, there are two ways to link MAL-based languages to each other: First, associations between different assets of languages can be created. In this case, additional attack steps need to be created to make use of these associations. Second, inheritance relations between the assets of the different languages can be established.

In our approach, we rely on the second way as it figured out that it requires less effort to keep the links when one of the languages is updated. In relation to the Purdue Enterprise Reference Architecture (PERA), the three languages can be situated on different level. coreLang refers mainly to the levels 3 and 4 representing the business IT. icsLang is mainly situated on the levels 2 (supervisory control) and 1 (control network itself), while sclLang includes level 1 and 0 (physical processes, sensors, and actuators). Consequently, we will link coreLang to icsLang and icsLang to sclLang as presented in Fig. 4.

To link coreLang to icsLang, we specify that the `IcsNetwork` is a specialized `Network` and `IcsAsset` is an `Application`. Due to this, the assets of both languages are linked to each other. Additionally, the attack steps need to be connected. Therefore, we overwrite *authenticate* that leads to the icsLang specific *authenticatedAccess*. Further integration is not needed.

Similarly, we proceed with linking icsLang and sclLang. We define that `Equipment` is a `Controller`, `IED` is a `SIS`, and `Server` is a `ControlServer`. To link the attack steps to each other, we overwrite *authenticatedAccess* that it leads to *execution* on `Equipment`.

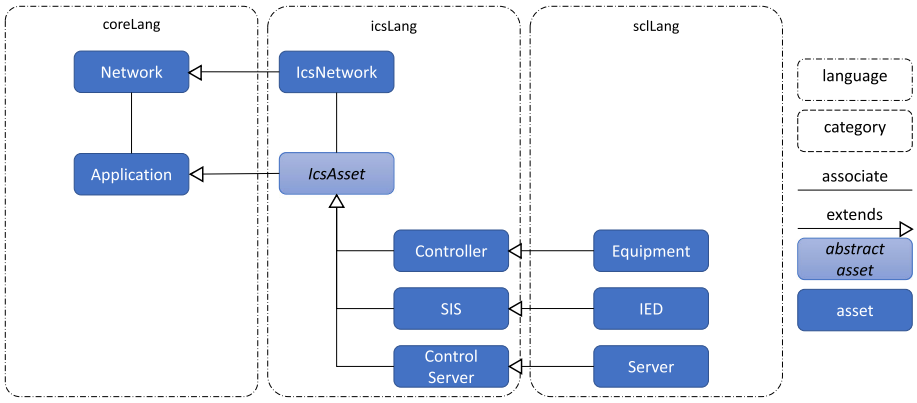

Fig. 4 Assets involved in linkage of languages

On the `IED`, we overwrite *access* that it leads to *communicate*. Same for `Server` where *authenticatedAccess* leads to *communicate*.

Demonstration

In 2015, an IT attack on the Ukrainian electric power grid switched off light for about 225,000 people (Defense Use Case 2016). This attack was the first documented successful cyber-attack on OT infrastructure affecting civilians. The attack was characterized by its coordinated and targeted approach to the critical infrastructure power supply. The attack involved a total of seven substations with 110 kV and 23 substations with 35 kV over a period of three hours. Manual interventions were needed to return to normal operations.

Figure 5 shows the simplified IT/OT architecture of a DSO created with powerLang[1] to simulate the behavior of an attacker. The attackers in the Ukrainian scenario used a spear-phishing attack on the office PC's of the network operators as initial attack vectors (Defense Use Case 2016). Remote access to the spear-phished users' PCs gained the attackers by using the malware BlackEnergy 3 (ThreatSTOP 2016). This initial foothold was used to lateral move in the network. Somehow the attackers were then able to capture virtual private network (VPN) credentials and, thus, got access to the OT network. Their main goal was to control the central SCADA systems of the DSO. Therefore, they took over control of the HMI that allowed them to access several controllers that operate switches, which led primarily to the blackout.

Additionally, the attackers performed supporting attacks to retain their access to the infrastructure and to hinder defensive responses. For example, firmware manipulation attacks were carried out against serial-to-Ethernet gateways in the process network, to the uninterruptible power supply. Further, KillDisk commands on operator workstations were performed to keep them down. This led to a refusal of service of these devices and increased the downtime and aggravation of the network rebuilding by the personnel (Defense Use Case 2016). However, these supporting attacks are not directly part of our simulations as they are reflected in the given probability distributions.

Given this architecture, we can perform the attack simulations in securiCAD. The first steps of the attacker are like already described. But then, the path splits up into three

[1] Both the model and the used powerLang version can be found under (Hacks et al. 2020)

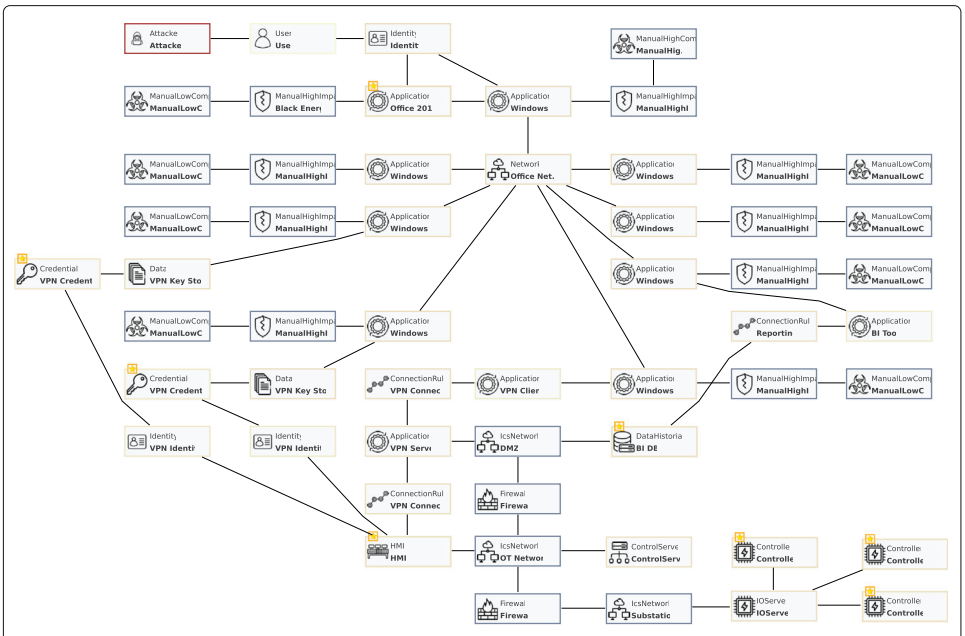

Fig. 5 Simplified IT/OT Architecture of a DSO

different ways. Two of them describe that the attacker uses different VPN-credentials to proceed to the OT. The third one follows another approach using the business intelligence (BI) interface of the data historian. From the data historian the attacker can choose different ways to moving the switches. This is also in line with our experience gained from discussions with practitioners who state that if attacker make it to the OT environment, they are almost free to do everything. Finally, the attacker gain access to the I/O server and then to the controller no matter which way they took. Apart from the attack path generation, securiCAD calculates also the most probable path, which is the one taken also in reality.

Discussion

The application of powerLang in the context of the known attack on the Ukrainian power grid unveils some research that are mainly related to the early stage of development of the language. First, we encounter that the amount of possible attack steps in icsLang is too large to understand how an attacker behaves. This is based on the fact that we took over the attack steps described on MITRE ATT&CK ICS. As the analysis of the attacker is not solely based on the simulation results, but also on a visual analysis of the security assessor, the amount of attack steps should be reduced to a human recognizable amount. Therefore, we will analyze the attack steps and join them in a second iteration where feasible.

Second, powerLang covers just general IT and OT aspects related to the needs of power grid operators. Aspects of power plants or smart metering are not included so far. Due to the modular structure of powerLang, these aspects can be added in future easily.

Third, we recognized that the extension mechanisms in MAL are not sufficient. To combine different, independently developed languages to each other, the source code needs to be copied from one repository to another. Therefore, we plan to packatize single

languages and to provide a package distribution similar to the way it is handled in Maven for Java libraries. Furthermore, we are reasoning about solutions to link languages to each other in a less tight way than using inheritance.

Fourth, a real world evaluation of powerLang is missing so far. However, it is already planned within the EnergyShield (2020) project and will be conducted in the Bulgarian demonstrator.

Conclusion

Within this article, we have developed powerLang that will enable power domain practitioners to model their IT/OT architecture and simulate the behavior of attackers within. This will allow them to identify possible choke points and improve infrastructure's security more efficient. To demonstrate this capability, we provided an exemplary model simulating the Ukrainian scenario.

However, there is still some work to be done. First, we solely tested our language for a scenario related to DSOs. But it is already planned to proof the language also in other environments and even in real-world settings. Second, many of attack steps are missing probability distributions that describe the expected effort to spend. These distributions are already substituted by the given probabilities inherited from IT-DSL. Nonetheless, future research needs to be conducted to gain more certain knowledge that needs to be spent on OT assets as this can be a significant difference compared to IT assets.

Apart from improvements of powerLang, this work has also shown that there are needs to improve mDSL's capabilities of integrating existing languages with each other to create new languages. Especially, a mechanism is needed for create the links between the languages and namespaces are needed to avoid naming issues. Furthermore, technical support is desirable to ease the distribution of languages like it is known from Maven for programming frameworks.

Acknowledgments
Not applicable.

Authors' contributions
SH administrated the overall writing of the article, developed icsLang, joined the different languages, contributed to coreLang, and created the demonstration example. SK created coreLang and contributed to the "coreLang" section. EL created sclLang and contributed to the "sclLang" section. ME and RL supervised the work and writing. All author(s) read and approved the final manuscript.

Funding
This work has received funding from the Swedish Civil Contingencies Agency through the research centre Resilient Information and Control Systems (RICS), European Union's H2020 research and innovation programme under the Grant Agreements no. 832907, the Swedish Centre for Smart Grids and Energy Storage (SweGRIDS), the Swedish Energy Agency, and the Swedish Governmental Agency for Innovation Systems (Vinnova). Open Access funding provided by Kungliga Tekniska Hogskolan.

Availability of data and materials
- Project name: powerLang
- Project home page: https://github.com/simonhacks/powerLang/
- Archived version: https://github.com/simonhacks/powerLang/releases/tag/v0.1
- Operating system(s): Platform independent
- Programming language: MAL
- Other requirements: securiCAD Professional 1.6.2
- License: Apache License 2.0
- Any restrictions to use by non-academics: none

Competing interests
The authors declare that they have no competing interests.

Received: 4 September 2020 Accepted: 26 October 2020
Published online: 25 November 2020

References

Alam M, Breu R, Hafner M (2007) Model-driven security engineering for trust management in sectet. JSW 2(1):47–59

Almorsy M, Grundy J (2014) Secdsvl: A domain-specific visual language to support enterprise security modelling. In: Software Engineering Conference (ASWEC), 2014 23rd Australian. IEEE, New York. pp 152–161

Baggett M (2008) Effectiveness of antivirus in detecting metasploit payloads. SANS Institute. https://pen-testing.sans.org/resources/papers/gcih/effectiveness-antivirus-detectingmetasploit-payloads-106563

Basin D, Clavel M, Egea M (2011) A decade of model-driven security. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies. ACM, New York. pp 1–10

CEN-CENELEC-ETSI SmartGridCoordinationGroup (2012) Smart grid reference architecture. https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf

Cherdantseva Y, Burnap P, Blyth A, Eden P, Jones K, Soulsby H, Stoddart K (2016) A review of cyber security risk assessment methods for scada systems. Comput Secur 56:1–27. https://doi.org/10.1016/j.cose.2015.09.009

Defense Use Case (2016) Analysis of the cyber attack on the ukrainian power grid. Electricity Information Sharing and Analysis Center (E-ISAC). https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

Ekstedt M, Johnson P, Lagerström R, Gorton D, Nydrén J, Shahzad K (2015) securiCAD by foreseeti: A CAD tool for enterprise cyber security management. In: Enterprise Distributed Object Computing Workshop (EDOCW), 2015 IEEE 19th International. IEEE, New York. pp 152–155

EnergyShield (2020) EnergyShield Homepage. https://energy-shield.eu. Accessed 01 Sept 2020

Franke U, Brynielsson J (2014) Cyber situational awareness – a systematic review of the literature. Comput Secur 46:18–31. https://doi.org/10.1016/j.cose.2014.06.008

Frigault M, Wang L, Singhal A, Jajodia S (2008) Measuring network security using dynamic bayesian network. In: Proc. of the 4th ACM Workshop on Quality of Protection. ACM, New York. pp 23–30

Habash RW, Groza V, Burr K (2013) Risk management framework for the power grid cyber-physical security. Curr J Appl Sci Technol 3(4):1070–1085

Hacks S, Hacks A, Katsikeas S, Klaer B, Lagerström R (2019) Creating meta attack language instances using archimate: Applied to electric power and energy system cases. In: 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC). IEEE, New York. pp 88–97

Hacks S, Katsikeas S, Ling E (2020) PowerLang repository. https://github.com/simonhacks/powerLang/. Accessed 01 Sept 2020

Holm H (2014) Signature based intrusion detection for zero-day attacks:(not) a closed chapter? In: 2014 47th Hawaii International Conference on System Sciences. IEEE, New York. pp 4895–4904

Holm H, Shahzad K, Buschle M, Ekstedt M (2015) P^2CySeMoL: Predictive, probabilistic cyber security modeling language. IEEE Trans Dependable Secure Comput 12(6):626–639. https://doi.org/10.1109/TDSC.2014.2382574

IEC (2018) Communication networks and systems for power utility automation - Part 6: Configuration description language for communication in electrical substations related to IEDs IEC 61850-6. https://webstore.ansi.org/Standards/IEC/IEC61850Eden2009

IEC Standard (2003) IEC 61850: Communication networks and systems in substations. Int. Electrotech. Commission, Geneva, Switzerland. https://www.academia.edu/34111821/Communication_networks_and_systems_in_substations_Part_1_Introduction_and_overview

Ingols K, Chu M, Lippmann R, Webster S, Boyer S (2009) Modeling modern network attacks and countermeasures using attack graphs. In: Computer Security Applications Conference, 2009. ACSAC'09. Annual. IEEE, New York. pp 117–126

Jiang Y, Jeusfeld M, Atif Y, Ding J, Brax C, Nero E (2018) A language and repository for cyber security of smart grids. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC). IEEE, New York. pp 164–170. https://doi.org/10.1109/EDOC.2018.00029

Johnson P, Lagerström R, Ekstedt M (2018) A meta language for threat modeling and attack simulations. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. ACM, New York. p 38

Jürjens J (2005) Secure systems development with UML. Springer, Berlin/Heidelberg

Katsikeas S, Hacks S, Johnson P, Ekstedt M, Lagerström R, Jacobsson J, Wällstedt M, Eliasson P (2020) An attack simulation language for the it domain. In: Graphical Models for Security. Springer, Berlin/Heidelberg

Katsikeas S, Johnson P, Hacks S, Lagerström R (2019) Probabilistic modeling and simulation of vehicular cyber attacks : An application of the meta attack language. In: Proceedings of the 5th International Conference on Information Systems Security and Privacy. SciTePress, Setúbal

Kordy B, Mauw S, Radomirović S, Schweitzer P (2010) Foundations of attack–defense trees. In: International Workshop on Formal Aspects in Security and Trust. Springer, Berlin, Heidelberg. pp 80–95

Kordy B, Piètre-Cambacédès L, Schweitzer P (2014) Dag-based attack and defense modeling: Don't miss the forest for the attack trees. Comput Sci Rev 13:1–38

Ling E (2020) sclLang repository. https://github.com/mal-lang/SCL-Lang. Accessed 13 Oct 2020

Liu Y, Ning P, Reiter MK (2011) False data injection attacks against state estimation in electric power grids. ACM Trans Inf Syst Secur (TISSEC) 14(1):13

Lund MS, Solhaug B, Stølen K (2010) Model-driven risk analysis: the CORAS approach. Springer, Berlin/Heidelberg

Masood R (2016) Assessment of cyber security challenges in nuclear power plants security incidents, threats, and initiatives. In: Technical Report - George Washington University. George Washington University, Washington, DC

Mauw S, Oostdijk M (2005) Foundations of attack trees. In: International Conference on Information Security and Cryptology. Springer, Berlin, Heidelberg. pp 186–198

MITRE (2020) MITRE ATT&CK. https://attack.mitre.org/. Accessed 01 Sept 2020

MITRE (2020) ATT&CK for Industrial Control Systems. https://collaborate.mitre.org/attackics/. Accessed 01 Sept 2020

Morikawa I, Yamaoka Y (2011) Threat tree templates to ease difficulties in threat modeling. In: 2011 14th International Conference on Network-Based Information Systems. IEEE, New York. pp 673–678. https://doi.org/10.1109/NBiS.2011.113

Noel S, Elder M, Jajodia S, Kalapa P, O'Hare S, Prole K (2009) Advances in topological vulnerability analysis. In: Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology. IEEE, New York. pp 124–129. https://doi.org/10.1109/CATCH.2009.19
Paja E, Dalpiaz F, Giorgini P (2015) Modelling and reasoning about security requirements in socio-technical systems. Data Knowl Eng 98:123–143
Petermann T, Bradke H, Lüllmann A, Poetzsch M, Riehm U (2011) Was Bei Einem Blackout Geschieht: Folgen Eines Langandauernden und Großflächigen Stromausfalls, vol. 662. Büro für Technikfolgen-Abschätzung, Berlin
Ramsbrock D, Berthier R, Cukier M (2007) Profiling attacker behavior following SSH compromises. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). IEEE, New York. pp 119–124. https://doi.org/10.1109/DSN.2007.76
Rosas-Casals M, Valverde S, Solé RV (2007) Topological vulnerability of the european power grid under errors and attacks. Int J Bifurcation Chaos 17(07):2465–2475
Schneier B (1999) Attack trees. Dr. Dobb's journal 24(12):21–29
Schneier S (2000) Lies: digital security in a networked world, vol. 21. John Wiley & Sons, New York
Sharifi A, Yamagata Y (2016) Principles and criteria for assessing urban energy resilience: A literature review. Renew Sust Energ Rev 60:1654–1677. https://doi.org/10.1016/j.rser.2016.03.028
Sommestad T, Hunstad A (2013) Intrusion detection and the role of the system administrator. Inf Manag Comput Secur 21(1):30–40
Stouffer K, Falco J, Scarfone K (2011) Guide to industrial control systems (ics) security. NIST Spec Publ 800(82):16–16
Ten C, Liu C, Manimaran G (2008) Vulnerability assessment of cybersecurity for scada systems. IEEE Trans Power Syst 23(4):1836–1846
ThreatSTOP (2016) Black Energy. https://www.threatstop.com/sites/default/files/threatstop_blackenergy.pdf. Accessed 15 May 2019
Wang W, Lu Z (2013) Cyber security in the smart grid: Survey and challenges. Comput Netw 57(5):1344–1371. https://doi.org/10.1016/j.comnet.2012.12.017
Wang JW, Rong LL (2009) Cascade-based attack vulnerability on the us power grid. Saf Sci 47(10):1332–1336
Williams L, Lippmann R, Ingols K (2008) GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool. In: Goodall JR, Conti G, Ma KL (eds). Visualization for Computer Security. VizSec 2008. Lecture Notes in Computer Science, vol 5210. Springer, Berlin
Witty RJ, Allan A, Enck J, Wagner R (2003) Identity and access management defined. Research Study SPA-21-3430, Gartner. https://www.gartner.com/en/documents/414388/identity-and-access-managementdefined
Xie P, Li JH, Ou X, Liu P, Levy R (2010) Using Bayesian networks for cyber security analysis. In: Dependable Systems and Networks (DSN), 2010 IEEE/IFIP Int. Conf. On. IEEE, New York. pp 211–220

Publisher's Note