Structured System Threat Modeling and Mitigation Analysis for Industrial Automation Systems

Roman Schlegel, Sebastian Obermeier, Johannes Schneider

ABB Corporate Research
Segelhofstr. 1K, Baden, Switzerland
firstname.lastname@ch.abb.com

***Abstract*—Industrial control systems are an important part of critical infrastructures and their uninterrupted operation is important for many aspects of society. In recent years these systems have come under more scrutiny, as reports about attacks on them have become more frequent. There is therefore a need to better secure them, and the first step to achieve this is to identify the threat landscape for such systems. This is typically done by creating a threat model of a system, enumerating the potential threats, and then devising mitigation options based on the discovered threats. However, many of the available threat modeling methods and tools target very specific systems (e.g., software components), and do not lend themselves well to evaluating diverse systems or abstract reference architectures of systems. In this paper we present a methodology for system threat modeling that addresses this gap, by enabling the modeling of a diverse range of systems and reference architectures of systems. Furthermore, the methodology provides additional functionality, such as guiding the user in the completion of a threat model and automatically detecting unmitigated threats in a system. In addition, our methodology also takes mitigation into account by modeling the security components in a system. We have implemented the methodology in a web-based tool, and also evaluated the tool on a reference architecture of a complex automation system, validating both the approach and the tool.**

I. Introduction

Systems used for monitoring and control within critical infrastructures such as energy networks or industrial SCADA (Supervisory Control and Data Acquisition) networks are increasingly complex, networked systems that often consist of hundreds of components from different manufacturers. In order to better understand security threats and mitigation options for these systems, methods such as brainstorming sessions or other meeting-based techniques are used. However, in order to create a more targeted output, a more formal method with a systematic approach should be considered to create a threat model.

Creating such a threat model for a device or system is an activity that helps to better understand the threats that are present. If a threat model is created during the development phase of a device or a system, it can identify changes that need to be made to the design or architecture to mitigate the discovered threats. If a threat model is created for a device or system that is already in operation, the results can help to prioritize any mitigation work. The positive impact on security of creating a threat model is arguably bigger if the threat model is created earlier in the life-cycle of a device or system, i.e., during the design and implementation phase. Engineering a device or system to be resistant against the attacks discovered in a threat model is likely to be much more cost-effective than retrofitting security or mitigating discovered threats at a later point in the life-cycle of a device or system.

There are different approaches for creating a threat model, i.e., *attacker-centric*, *asset-centric*, or *system-centric*. They mainly differ in how a device or system is evaluated, namely:

Attacker-centric: In this approach the threat model is created by taking an attacker's view, and trying to determine how an attacker would try to break the system.

Asset-centric: This approach of creating a threat model looks at the assets in a system that need to be protected (e.g., valuable information, critical components, etc.) and determines the threats to these assets.

System-centric: In a system-centric approach, a device or system is examined component by component, and for each component the threats are assessed.

Independent of the approach taken to create a threat model, another important decision is *how* to create the threat model, i.e., which methodology and which tools should be used. There are many different tools and methods available to create threat models, e.g., [1], and they differ significantly in their sophistication and their specialization towards analyzing a certain type of system, e.g., some tools are targeted specifically towards modeling software components. There is a wide range of tools, from a simple text document with unstructured text, to the use of mind mapping software for threat models, to very specialized tools that provide integrated analysis functions on the threat model. There is therefore a spectrum of tools, from very general tools on one end to very specific tools on the other end, with different advantages and disadvantages depending on where in the spectrum such a tool is located.

Specifically, looking at the two extremes of this spectrum (very general and very specific), there are a number of advantages and disadvantages to both types of tools. General tools do not constrain the threat model to conform to a certain structure or data model, while specific tools typically have an underlying data model that requires the threat model data to conform to a certain structure. This results in the following strengths and weaknesses for the two different types of tools.

General Tools. General tools have the following advantages:

- flexibility, they can be used to represent almost everything
- very little constraints on how the information needs to be structured

978-1-4799-6649-3/15/$31.00 ©2015 IEEE

However, their strengths are at the same time also the reason for their weaknesses:

- the lack of constraints leads to a lack of structure within the data
- because of the lack of structure, it is hard to re-use data in a threat model, for example for a similar threat model, or to automatically deduce additional information

On the other end of the spectrum are very specific tools, that are typically focused on creating a threat model for a specific type of device or system (e.g., software):

Specific Tools. The specialization of such tools leads to the following advantages:

- the underlying data model provides the threat model with a well-defined structure
- the structure in the data can enable the tool to run automated analysis algorithms on the data

But the specialization of the tools also leads to disadvantages:

- the data model of the tool can restrict the applications of the tool
- such tools are therefore often only able to analyze specific devices or systems (e.g., only software components)

In our experience, many of the tools fall mostly in one of the two categories (i.e., general or specific), with relatively few tools available that try to strike a balance between generality and specificity. Such a balance, however, would enable a tool to be used for creating threat models for different types of devices or system, while at the same time providing enough structure for the data so that it can be analyzed automatically using different analysis functions.

Another advantage of more balanced tools is that they can be used to more easily model reference architectures of a system, instead of having to model specific instances of a system. In our case we needed to create a threat model for a reference architecture of a system, such that the resulting threat model would also be applicable to specific instances of such a system. However, the existing tools we found were either too general, and would not allow an efficient re-use of data, or too specific, in that they could not reasonably be applied to all the different components in a reference architecture.

We therefore developed a new threat modeling methodology and a corresponding tool that tries to strike a balance between generality and specificity. The methodology is specific enough to enforce a basic structure on the data and allow some automatic processing and analysis of the data, but still general enough as not to be too restrictive in the types of systems that can be modeled. In addition, the methodology supports automatically analyzing a threat model for unmitigated threats, and at the same time also suggesting security controls that could mitigate the identified threats.

The targeted audience for our threat modeling methodology and the corresponding tool are security professionals, i.e., sufficient security knowledge is a prerequisite for the use of our methodology. This is in contrast to other tools and methodologies that try to make threat modeling or security assessments also accessible to non-experts (e.g. [2]). However, creating a threat model from scratch using our methodology requires knowledge of the different threats and how they apply to the different components.

The rest of this paper is organized as follows: Section II introduces our system threat modeling methodology and gives an overview of the tool we developed that implements this methodology. In Section III we give an evaluation of our methodology and the tool, followed by an overview of related work in Section IV. Finally, we conclude in Section V.

II. System Threat Modeling

In this section we first describe our system threat modeling methodology and then give an overview of our tool that implements this methodology.

A. Methodology

To address our need for a methodology that was general enough but still provided enough structure to effectively work with the data of a threat model, we developed a data model and defined relationships within the data model to capture the important elements of a threat model. However, our methodology goes one step further than a regular threat model, by not only modeling components and threats within a system, but also the security controls that can mitigate threats.

Data model
The data model (shown in Figure 1) used to describe the contents of a threat model contains 4 different elements, each of which represents an object type in the threat model. These 4 elements are:

Component: These are the components of the system for which the threat model is created. As an example, if a threat model of a computer network is created, the components could be servers, desktop computers, network switches, etc.

Threat: These are the threats that have been identified, and which apply to the components in the system. An example threat would be “malware”, which would apply to a desktop computer.

Impact: An impact is the combination of a threat and a component. For example, the threat “malware” applying to a component “desktop computer” produces a certain impact, namely, the compromise of the desktop computer.

Security Control: Introducing the notion of security controls into the data model extends our threat models from pure threat models to models that also contain information on the mitigation of threats. An example for a security control could be “antivirus”, which mitigates the threat “malware”.

Figure 1 shows the relations between the 4 different elements. Specifically, components and threats are linked by impacts, i.e., a threat applying to a component produces an impact. In addition, security controls can mitigate threats, and components can implement security controls, which then mitigate threats that apply to this component. This data model with its relationships enables the application of analysis algorithms on the data, making it possible to derive additional information about the objects from the data in the threat model.

Furthermore, there are additional attributes that are linked to some of the elements in the threat model. The “threat” object contains an attribute “likelihood”, that gives a rough estimate on how likely it is that a threat would be realized. For the sake

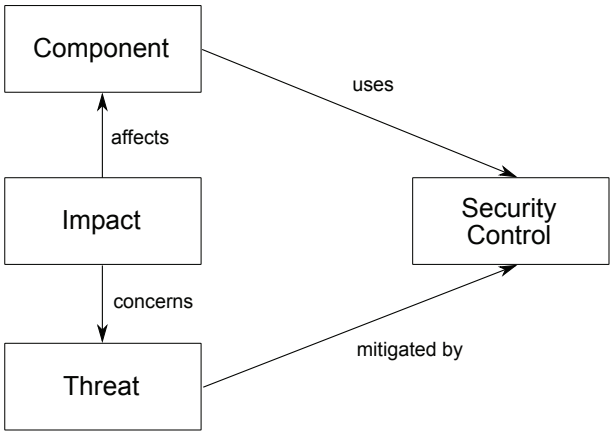

Fig. 1. The data model of our threat modeling methodology.

of simplicity this is expressed as a value from 1 to 3 (or low to high), with 1 denoting a small likelihood, and 3 denoting a high likelihood. Determining the likelihood is of course often subject to discussion and interpretation, but the idea is to provide additional (if approximate) information in the model, that can be used when analyzing the model. Likewise, the "impact" object also contains an additional attribute "severity", which expresses the severity of an impact in terms of the effect on security. This is again expressed as a simple 3-point scale, with 1 denoting a small impact or severity, and 3 denoting a large impact or severity. We purposely did not opt for a more detailed scale for describing likelihood and severity (e.g., a percentage from 0% to 100%), as giving any more precise values feigns an accuracy that simply does not exist. Furthermore, even a simple scale as described above allows automatic processing of a model (e.g., to determine an ordering for unmitigated threats).

This data model lends itself very well to being stored in a relational database, which also makes the implementation of a corresponding tool relatively straightforward.

Threat Modeling

Given the data model described above, a threat model can be created by mapping the objects in the threat model to the data model, and defining the relationships between the objects. While there is no specific order for the steps that needs to be followed, we found the following order to be efficient:

1) Populate the threat model with a list of threats. These threats can either be relatively general, or more specific threats targeted at the system being analyzed. They can also be refined later in the process and new threats can be added at any point during the threat modeling activity.
2) Add the list of possible security controls to the threat model and correlate them with the threats, i.e., indicate which security control can mitigate which threat.
3) Add the components in the system to the threat model. For each component, indicate which security controls it currently implements, if any.
4) Correlate the components and threats by adding an impact for each threat that applies to a component.
5) Continue to refine the model by adding newly discovered threats, or additional security controls. Correlate them with the components and add any new impacts if necessary.

The end result of the process is a model of the system being investigated, with the appropriate objects being linked through the different relationships in the data model. This then constitutes the threat model of the system.

B. System Threat Modeling Tool

To evaluate the methodology we have created a tool that can be used to perform threat modeling for systems. This tool has the following features:

- web-based interface to allow for easy collaboration within a team
- relational database back-end to store all information about the threat model
- relational data model that provides a structure for the threat modeling data
- easy export and import of a complete threat model or of parts of a threat model
- analysis algorithm that can automatically determine unmitigated threats and provide suggestions for mitigation options in the form of security controls
- multi-instance operation, where one installation of the tool can provide access to separate and independent threat model instances

Many of the features of this threat modeling tool stem from the fact that all data is stored as entities in a relational database, and the underlying data model ensures that this mapping to a database is trivial.

User Interface

The user interface of the threat modeling tool is web-based, i.e., the threat model can be edited in its entirety through a web browser over the network. The tool provides two distinct views, due to the underlying software framework (the Django web framework[1]):

Explorer View: This view allows the user to explore the threat model and examine the information associated with all objects, presenting all the data in the threat model in a structured way.

Administration View: The administration view is where the threat model can be edited, e.g., new components, threats, etc., can be added, edited and deleted.

The tool also supports access rights, either requiring a user to be logged-in before changes can be made to a threat model, while the model can be explored freely, or requiring a user to log-in for all operations, even just browsing the threat model.

Guided Threat Model Completion

In order to help the user complete the threat model and ensure that all relevant combinations of components and threats have been considered, the tool automatically notifies the user whenever there are any impacts (i.e., a threat applying to a component) that have not been defined yet in the model. This is achieved by having the user indicate for each new component which threats potentially apply to a component. This makes the workflow of adding all relevant impacts for a component much more efficient, as the tool will present a list with all potentially applicable threats where the corresponding impact is still missing. A simple click will then allow a user to add the corresponding impact (Figure 2).

[1] https://www.djangoproject.com/

Fig. 2. A list of threats that potentially apply to a component, but for which no impact has been defined yet. A click on one of the links will directly open the appropriate interface to add the corresponding impact.

Impact Matrix

Together with the guided threat model completion described above, the impact matrix is another part of the tool that gives an overview over the state of the threat model. By plotting all components and threats in a two-dimensional matrix and marking the already existing impacts (i.e., the threats that apply to a component), the completeness of a threat model can easily be verified (see Figure 3).

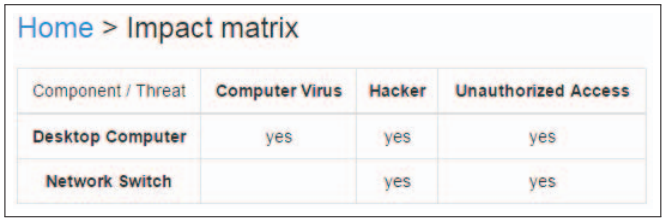

Fig. 3. The impact matrix shows for which combinations of threats and components an impact has been created in the threat model.

Export / Import of Data

To enable the re-use of data in a threat model, the tool has a functionality to export (and import) the data contained in a threat model. The complete threat model can either be exported into a hyperlinked Word document that can be used as the basis for a report, or it can be exported in JSON format to be be imported again into another threat model instance. One common use-case of the JSON export/import is to share information between threat models, such as for example the list of threats. A threat list has therefore only to be created once, and can then be re-used across multiple, different threat models.

Analysis Algorithms

The advantage of using a relational data model and storing all objects of the threat model in a relational database is that it enables the use of analysis algorithms on the data. With the relations between the objects (i.e., threats apply to components, security controls mitigate threats, and components can implement security controls) the interactions between objects can be examined. As an example, the threat "malware" applies to a component "desktop computer" and can be mitigated by the security control "antivirus". An analysis algorithm can now evaluate the threat model and determine which threats have been mitigated, by combining the information which threats can be mitigated by which security controls and which security controls are implemented by each component. To continue the example, if the "desktop computer" does not implement the security control "antivirus", then the analysis algorithm can determine that the desktop computer is in fact still vulnerable to malware.

This can be extended even further by using an analysis algorithm that not only finds the security gaps in the threat model (i.e., unmitigated threats), but also directly suggests mitigation options. If the algorithm determines that malware is an unmitigated threat for the desktop computer, it can use the information contained in the model and go through the list of security controls to see if any of the controls can mitigate the threat malware, and if it finds one (e.g., antivirus), directly suggest it as a possible mitigation option. Figure 4 shows a list with unmitigated threats in an example threat model. For each threat, the system automatically suggests mitigation options. This list is dynamically generated, and any updates in the threat model (e.g., adding a security control to a component) are reflected immediately.

Fig. 4. A list of unmitigated threats in an example threat model, together with suggested mitigation options.

III. Evaluation

We have evaluated our methodology and the corresponding tool by applying it to a real-world industrial application: a substation automation system (SAS). The SAS monitors, controls and protects power equipment in a switchyard facility, and is described and standardized in the IEC 61850 standard [3].

A modern SAS as shown in Figure 5 is organized by utilizing two communication buses: a station bus and a process bus. The station bus interconnects the station level devices such as station computer, gateway, etc., with bay-level intelligent electronic devices (IEDs) that perform bay protection and control functions. The station bus typically carries Generic Object Oriented Substation Event (GOOSE) and TCP/UDP traffic such as MMS, SNTP, or FTP.

The process bus interconnects the IEDs with the Merging Units (MUs) which interface to the physical process and publish digital values corresponding to the analog current and voltage of a bay to their respective IEDs. The process bus typically carries Sampled Values (SV) and GOOSE traffic on Layer 2 of the ISO/OSI model. GOOSE messages are used for reliable multicast of status and control information. SV messages are used for sending periodic updates (unicast or multicast) of current and voltage samples.

The main motivation for creating a threat model for an SAS is to formalize why certain security features have been included, which threats they mitigate, and which additional security features would be beneficial to further improve se-

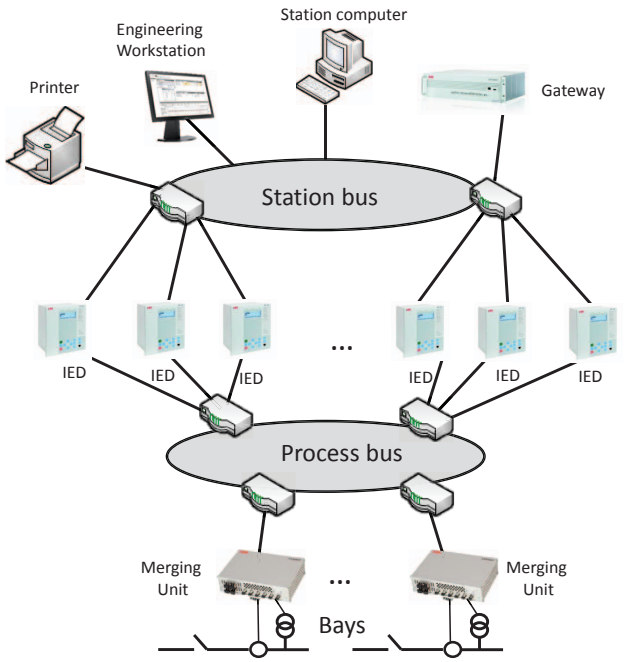

Fig. 5. Substation Automation System

curity. The challenge of creating a threat model for an SAS system is the "state explosion": in a typical instantiation of an SAS, there are more than 20 individual assets, a multitude of communication channels that utilize a plethora of protocols, and even devices of the same type but of different vendors for redundancy purposes. Existing approaches such as STRIDE focus on a guided exploration of threats for a relatively small piece of software. In an SAS system, each device also has its own threat model, which contributes to the overall system. However, simply performing a device-level threat model for each device does not give an accurate overview of the threat landscape of the whole system which also takes interaction between the devices into account. So while the general threats to an SAS are known, they cannot be accurately captured in device-level threat models.

Using our methodology and the corresponding tool, we started by adding all components of an SAS including a short description. In parallel, we added 20 generic threats (e.g., buffer overflow, data flooding, etc.) and 18 generic security controls (anti-virus, robustness testing, configuration review, etc.). Because each threat has a different impact on different components, we linked components and threats: a threat that affects a component causes a certain impact, which is described and ranked with a corresponding severity (i.e., high/medium/low). We also reflected which components already implement which security features by linking each component with the respective security controls.

We compared the threat modeling done manually to the threat modeling done with our tool on a reference architecture of an SAS. The formalism employed in the tool allowed us to get a complete and detailed overview of the security state of the SAS and automatically derive recommendations for adding additional security controls. In total, the tool was able to suggest more than 10 security controls that can be added for improved security.

To summarize our evaluation, the methodology provided a structured and efficient way of performing the threat modeling. Furthermore, the underlying data model allows to automatically derive the security state of a system, provide suggestions for additional security controls, and enables the re-use of lists of threats and security controls.

IV. Related Work

A. Methodologies

Several threat modeling approaches have been proposed in the current literature [4]–[9]. Attack trees [4] represent security of a system with respect to a multitude of attacks, frequently sharing some commonalities. Attacks on a system are shown as a tree with the goal as the root node and the path to the root describing the steps of an attack, i.e., each node on the path specifies a step of the entire attack. Several attributes can be assigned to nodes such as the cost of executing the step or the feasibility of the attack. This allows for further analysis answering questions like “What if an attacker has a budget of 100000 dollars?”. Attack trees have been used to model vulnerabilities in SCADA systems [10], which uses a quantitative approach to assess certain vulnerabilities. For example, they create a password model to estimate a possible compromise, which utilizes the number of intrusion attempts and the number of observed records to estimate a transition probability. In contrast, we focus on deriving a threat modeling approach with tool support.

The DREAD methodology [5] classifies the amount of risk of a threat according to damage, reproducibility, exploitability, affected users and discoverability. The STRIDE model (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) provides a categorization of threats [5], [11]. Microsoft also developed a card game [12] to teach threat modeling according to STRIDE. In the tools section we elaborate on the usefulness of STRIDE in our context.

The CORAS methodology [6] uses the unified modeling language (UML) and the unified process (UP) as the modeling language for risk management of IT systems. It is based on the Australian Risk Management standard [13] and allows for several risk assessment methodologies such as hazard and operability study (HazOp) [14] and fault tree analysis [15]. In the tools section we discuss in more detail the suitability of the methodology.

Agile methods have also been used for threat modeling [7]. A method in [7] is based on so called “abuser” stories describing ways an attacker might compromise the system. The process consists of three steps: identifying threats using STRIDE, identifying mitigation options and, finally, testing.

Other work [8] introduces threat modeling based on attack paths with emphasis on the impact of an attack. An attack path as used by [8] shows the locations as well as vulnerabilities of an attack. Abuser stories should ideally be developed together with the customer. This is not always suitable in our setting, since customers are typically not aware of the complexity of the system and the many threats that arise because of it. Furthermore, industrial systems are typically not developed in an agile manner like in software engineering, since iteratively changing a physical system is very costly. Therefore, overall we believe that this approach is not suitable in our context.

Employing Fuzzy logic to assess threats defined by the STRIDE model has been suggested in [9]. The idea is to transform linguistic qualitative variables into numbers, e.g., "low probability for spoofing" gets 0.3. The outcome of this first step serves as an input for rule evaluation, followed by aggregation and finally defuzzification. Rules are of the form, e.g., "(Spoofing is low) and (Tampering is low) and ...". Though a numerical evaluation might be desirable, we did not consider numerical estimates of probabilities, since they are difficult to obtain and unreliable. In contrast to our methodology, the approach [9] focuses only on threats at the system level and does not allow to model mitigations. It also does not allow to specify which components are impacted by a threat in a straightforward manner.

Threat modeling is the subject of several standards [13], [16]. The Australian/New Zealand standard [13] provides a general risk management methodology consisting of five steps, establish context, identify, analyze, evaluate and treat risks. The ISO 17799 [16] also provides high level guidelines for ensuring confidentiality, integrity and availability of information assets through implementing security controls. It strongly emphasis organizational concerns such as employing a security officer and security policies.

B. Tools

There exist a variety of tools for threat modeling. The CORAS methodology [6] is also implemented as a tool [17] as well as the fuzzy logic approach to threat modeling [9]. Figure 6 shows a screenshot of the CORAS tool [17]. The model shown contains the following elements: threats (depicted as devils with a triangle), vulnerabilities (open locks), risk (triangles), unwanted incidents (stars) and assets (money bag). Additional elements exist. We believe that our approach captures real world systems with less (modeling) elements and thus reduced complexity. Though the graphical user interface might seem appealing, we believe that our matrix presentation is better suited for a larger number of threats and leads to a better overview, which allows more easily to spot missing links among, e.g., threats and components. Furthermore, laying out the elements graphically must be done manually, which can become burdensome for larger models.

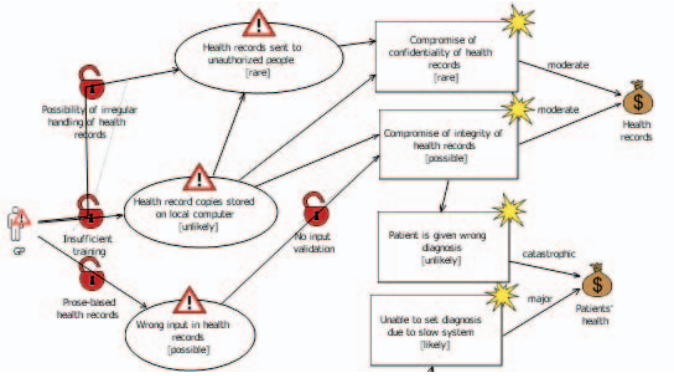

Fig. 6. Screenshot of the CORAS tool [17] for three threats. Despite the graphical structure the key elements cannot be grasped on first sight.

The Microsoft Software Development Lifecycle (SDL) Threat Modeling Tool [18]–[20] was designed for non-security experts. The tool allows to visually model the architecture of the system including its users and relationships among components as well as to specify the trust boundary. For each component threats are classified according to the STRIDE model. The tool comes with high level, abstract suggestions for mitigation options and threats. Thus, to a large extent the user must enter threats and mitigation options manually. The modeling approach differs, since mitigations are tied to threats in a 1:1 to fashion, i.e., for each threat in the STRIDE model one must enter possible impacts and solutions in a text field, whereas we model mitigations and impacts as entities. For instance, our methodology allows to model scenarios, where one mitigation deals with several threats (at least partially).

Microsoft has also developed a threat and analysis modeling process and tool (TAM) [21], [22] targeted towards application threat modeling. It is use-case driven and it puts emphasis on users, their roles and (access) rights. In contrast to Microsoft's STRIDE model, threats are classified into three categories, i.e., confidentiality, integrity and availability. Use-cases also highlight the dependencies among components. We found the tool to be too specific for application modeling, e.g., it was not clear how to account for spoofing of a GPS signal without workarounds or abusing nomenclature. Furthermore, we did not put the same emphasis on use-cases as Microsoft's tool.

The open source tool SeaMonster [23] allows to visually model in three different viewpoints what causes a vulnerability, how this poses a threat to the system and how to mitigate the threat. In our opinion the tool is not ideal for system modeling, since it does not come with the concept of components. For example, it is not straightforward without workarounds to cleanly model (e.g., without manual threat duplication) that a denial of service attack can occur on a central company wide authentication server as well as on a computer used for monitoring in a substation. Furthermore, while the tool excels in a detailed modeling approach of each vulnerability, it lacks an overview, which makes it more susceptible to missing a certain threat and mitigation or component threat scenarios. Therefore, from our point of view it is less suitable in the context of complex industrial systems.

The Common Vulnerability Scoring System (CVSS) [24], [25] is a quantitative modeling approach for IT vulnerabilities. For each vulnerability scores can be defined with respect to exploitability, impact etc. that yield the overall score. The tool is rather limited in the sense that it is not possible to explicitly enter threats or mitigations, i.e., the tool only deals with computing a vulnerability score for a system.

The "Trike" modeling tool [26], [27] takes into account data models, use cases, connections, protocols and security objectives. Data is entered in a spreadsheet type manner. The modeling approach is targeted more towards IT infrastructure of non-industrial companies than towards automation systems. For example, it heavily focuses on actors, assets, their intended actions, data model, etc., and each of these aspects has its own sheet that needs to be filled. For automation systems the number of actors, their possible actions as well as the assets are typically more limited. Complexity arises due to a large number of diverse components and their interactions, ranging from special sensors to conventional desktop computers in a distributed setting, e.g., for substation systems. This exceeds the diversity of non-industrial IT systems. Therefore, we found

the approach to be too specific for our intentions.

Commercial tools include ThreatModeler which employs data elements, roles and components drawing from several attack libraries [28]. It outputs attack trees with the component as a root, violated requirements as the first level, followed by layers consisting of threats and attacks. Another commercial tool is "Corporate Threat Modeller" [29]. It allows to specify risks on a chosen interface from a certain location by a type of user including an impact and a likelihood. Both commercial tools [28], [29] target primarily software systems and therefore face similar challenges as described for "Trike" [26], [27] above when applied to industrial systems.

The ISO 17799 RAT tool is based on the ISO 17799 standard [30]. Unfortunately, at the time of writing the tool was not available in English.

V. Conclusions

In this paper we have presented a threat modeling methodology that fills a gap in existing tools, which are either very specific (and hence only applicable to a very narrow field), or very general (making the data too unstructured as to be usable for any automated processing). Our methodology provides a compromise between too specific and too general, while still allowing the threat model to be as detailed or as coarse as required, without restricting the threat model either way. Furthermore, the methodology lends itself well to being applied to abstract reference architectures of systems. In addition to the methodology itself, we have created a tool that implements this methodology and provides a number of additional features, such as automatically suggesting mitigation options, as well as guided completion of a threat model and easy export and import of data. We have also evaluated the tool by applying it to a reference architecture of a substation automation system, and it helped to direct the focus of the development teams towards areas that could most profit from further strengthening security.

Concerning future work, one extension would be to also model the communication links between components and take this into account when analyzing the model. Other extensions could include additional analysis algorithms, or the possibility to have a more fine-grained object model that would also allow hierarchies of components.

References

[1] H. Q. Nguyen, F. Köster, M. Klaas, W. Brenner, S. Obermeier, and M. Brändle, "Tool support for achieving qualitative security assessments of critical infrastructures - the ESSAF framework for structured qualitative analysis," in *SECRYPT 2009, Proceedings of the International Conference on Security and Cryptography, Milan, Italy, July 7-10, 2009, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, E. Fernández-Medina, M. Malek, and J. Hernando, Eds. INSTICC Press, 2009, pp. 297–304.

[2] A. Hristova, R. Schlegel, and S. Obermeier, "Security assessment methodology for industrial control system products," in *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, June 2014, pp. 264–269.

[3] International Electrotechnical Commission, IEC TC57, "IEC 61850."

[4] B. Schneier, "Attack trees," *Dr. Dobb's journal*, vol. 24, no. 12, pp. 21–29, 1999.

[5] D. LeBlanc and M. Howard, *Writing secure code*. Pearson Education, 2002.

[6] F. den Braber, T. Dimitrakos, B. A. Gran, M. S. Lund, K. Stølen, and J. Ø. Aagedal, "The coras methodology: model-based risk assessment using uml and up," *UML and the Unified Process*, pp. 332–357, 2003.

[7] J. Peeters, "Agile security requirements engineering," in *Symposium on Requirements Engineering for Information Security*, 2005.

[8] Y. Chen, B. Boehm, and L. Sheppard, "Value driven security threat modeling based on attack path analysis," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*. IEEE, 2007, pp. 280a–280a.

[9] A. Sodiya, S. Onashoga, and B. Oladunjoye, "Threat modeling using fuzzy logic paradigm," *Informing Science: International Journal of an Emerging Transdiscipline*, vol. 4, no. 1, pp. 53–61, 2007.

[10] C.-W. Ten, C.-C. Liu, and G. Manimaran, "Vulnerability assessment of cybersecurity for scada systems," *Power Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 1836–1846, 2008.

[11] Microsoft, "http://msdn.microsoft.com/library/ms954176.aspx," http://msdn.microsoft.com/library/ms954176.aspx, [Online; accessed 2014-11-27].

[12] Microsoft, "Elevation of Privilege Card Game," http://www.microsoft.com/security/sdl/adopt/eop.aspx, [Online; accessed 2014-11-26].

[13] D. Cooper, "The australian and new zealand standard on risk management, as/nzs 4360: 2004," *Tutorial Notes: Broadleaf Capital International Pty Ltd*, 2004.

[14] J. Dunjó, V. Fthenakis, J. A. Vílchez, and J. Arnaldos, "Hazard and operability (hazop) analysis. a literature review," *Journal of hazardous materials*, vol. 173, no. 1, pp. 19–32, 2010.

[15] C. A. Ericson and C. Ll, "Fault tree analysis," in *System Safety Conference, Orlando, Florida*, 1999, pp. 1–9.

[16] M. Kenning, "Security management standard - iso 17799/bs 7799," *BT Technology Journal*, vol. 19, no. 3, pp. 132–136, 2001.

[17] "CORAS Risk Assessment Platform," http://sourceforge.net/projects/coras/, 2007, [Online; accessed 2014-11-28].

[18] Microsoft, "SDL Threat Modeling Tool," http://msdn.microsoft.com/en-us/security/dd206731.aspx, 2008, [Online; accessed 2014-11-27].

[19] B. Potter, "Microsoft sdl threat modelling tool," *Network Security*, vol. 2009, no. 1, pp. 15–18, 2009.

[20] A. Shostack, "Experiences threat modeling at microsoft," in *Modeling Security Workshop. Dept. of Computing, Lancaster University, UK*, 2008.

[21] J. A. Ingalsbe, L. Kunimatsu, T. Baeten, and N. R. Mead, "Threat modeling: diving into the deep end," *Software, IEEE*, vol. 25, no. 1, pp. 28–34, 2008.

[22] Microsoft, "Threat and Analysis Modeling Process," http://msdn.microsoft.com/en-us/security/aa570413, [Online; accessed 2014-11-27].

[23] P. H. Meland, D. G. Spampinato, E. Hagen, E. T. Baadshaug, K.-M. Krister, and K. S. Velle, "Seamonster: Providing tool support for security modeling," *Norsk informasjonssikkerhetskonferanse, NISK*, 2008.

[24] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright, and S. Romanosky, "Cvss: A common vulnerability scoring system," *National Infrastructure Advisory Council (NIAC)*, 2004.

[25] NIST, "Common Vulnerability Scoring System Version 2 Calculator," http://nvd.nist.gov/cvss.cfm?calculator&version=2/, 2007, [Online; accessed 2014-11-26].

[26] P. Saitta, B. Larcom, and M. Eddington, "Trike v. 1 methodology document [draft]," *URL: http://dymaxion.org/trike/Trike_v1_Methodology_Document-draft.pdf*, 2005.

[27] ——, "Trike," http://octotrike.org/, 2005, [Online; accessed 2014-11-27].

[28] MyAppSecurity, "ThreatModeler," http://myappsecurity.com/threatmodeler/, [Online; accessed 2014-11-26].

[29] SensePost, "Corporate Threat Modeller," http://research.sensepost.com/tools/management/ctm, [Online; accessed 2014-11-28].

[30] "RAT ISo 1779 Risk Analysis ToolKit," http://ratiso17799.sourceforge.net/, 2007, [Online; accessed 2014-11-28].