Designing and Integrating IEC 62443 Compliant Threat Analysis

Conference Paper · September 2019

DOI: 10.1007/978-3-030-28005-5_5

Markus Fockel
Fraunhofer IEM
22 PUBLICATIONS **78** CITATIONS

Masud Fazal-Baqaie
Next Data Service AG
36 PUBLICATIONS **108** CITATIONS

Designing and Integrating IEC 62443 Compliant Threat Analysis

Markus Fockel[1][0000−0002−1269−0702], Sven Merschjohann[1][0000−0001−9133−3350], Masud Fazal-Baqaie[1][0000−0002−0981−126X], Torsten Förder[2], Stefan Hausmann[3], and Boris Waldeck[3]

[1] Software Engineering and IT Security, Fraunhofer IEM, Paderborn, Germany
{markus.fockel,sven.merschjohann,masud.fazal-baqaie}@iem.fraunhofer.de
[2] Phoenix Contact Software GmbH, Lemgo, Germany
tfoerder@phoenixcontact.com
[3] Phoenix Contact Electronics GmbH, Bad Pyrmont, Germany
{shausmann,bwaldeck}@phoenixcontact.com

Abstract. Cybersecurity gains more and more attention as the number of security incidents rises. In order to strengthen the security of products within the industrial automation domain, the novel standard IEC 62443 prescribes security practices throughout the development lifecycle that improve the security of the resulting product. However, implementing and integrating concrete security practices into the existing development processes is challenging, as best practices for the automation domain are still missing. Hence, in this paper we present our implementation of a standard compliant threat analysis for the development process of the industrial control systems manufacturer Phoenix Contact. Phoenix Contact was successfully certified for its compliance with IEC 62443. We illustrate the requirements of the standard, the resulting threat analysis process, and its tight integration into the existing development process and its tools.

Keywords: Threat analysis · STRIDE · IEC 62443 · security by design.

1 Introduction

The degree of software within products and services is growing across all domains [10]. In addition, the usage of standard internet and computer technologies for the connectability of devices, e.g., to the Internet, is rising. Both trends lead to increasing numbers of security incidents that draw more and more public attention to the topic of cybersecurity [7]. The growing risks due to insufficient cybersecurity have led to a number of measures in order to professionally monitor and mitigate, but also to prevent security incidents. As safety cannot be guaranteed without taking security into account, some industries establish security standards that enforce the proper use of security practices (e.g., the automotive domain [6, 5]). This is also the case for the industrial automation domain for which the novel standard IEC 62443 [3, 4] prescribes product and process requirements for security. Hence, with this standard, organizations receive a holistic set of guiding principles to implement security practices throughout the complete lifecycle of their products. However, the standard is implementation-agnostic and describes *what* needs to be implemented, but not *how*. Therefore, organizations have to come up with concrete security practices that conform to the standard. However, as it is a novel standard, there are no established best practices for the industrial automation domain. Thus, companies struggle with implementing concrete security practices into their existing development processes such that it meets the requirements of the standard and that they can get certified for compliance with IEC 62443.

Phoenix Contact is one of the companies that faced this challenge. The company is a manufacturer of industrial automation, interconnection, and interface solutions. Among other products, the company produces programmable logic controllers (PLC) that are a core component of industrial control systems within the industrial automation domain. The company is now among the first ones that is certified with respect to the standard IEC 62443. In this paper, we focus on threat analysis as a security practice early in the development lifecycle. Using threat analysis as an example, we illustrate, how we designed IEC 62443 compliant security practices and how we implemented them into the existing development processes for PLCs at Phoenix Contact.

In agreement with the principles for the SPI manifesto [8] for software process improvement, the resulting process aims not only at enabling certification according to the standard. Equally important was to change the existing development process, such that the employees raise their awareness for security and indeed implement security measures into their daily practice, ultimately lowering risks and making the business more successful. Especially, the people factor was accounted for during requirements analysis in order to come up with an accepted and effective process improvement.

This paper is structured as follows. In Sec. 2, we give an overview of the security practices along the development process and put threat analysis into the context of the other security practices. All the practices together make up a secure development lifecycle as required by IEC 62443. In Sec. 3, we describe the initial situation and requirements of the threat analysis for Phoenix Contact as well as the designed solution that was integrated into the existing development process. We also discuss how it fulfills the stated requirements. In Sec. 4, we present lessons learned from the project with findings that we like to highlight. We finally conclude the paper in Sec. 5 with a summary and outlook.

2 Overview of the Secure Development Lifecycle

In order to systematically develop secure products, security must be emphasized throughout the whole software lifecycle, such that the results are secure by design. Thus, every phase of a common software lifecycle has to be enhanced with security practices. This leads to a secure development lifecycle as illustrated in Fig. 1.

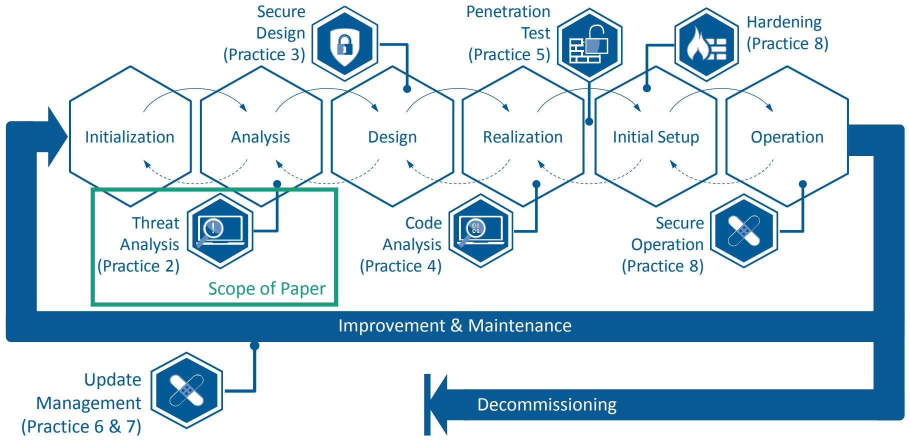
Secure Design (Practice 3)
Penetration Test (Practice 5)
Hardening (Practice 8)
Initialization
Analysis
Design
Realization
Initial Setup
Operation
Threat Analysis (Practice 2)
Scope of Paper
Code Analysis (Practice 4)
Secure Operation (Practice 8)
Improvement & Maintenance
Update Management (Practice 6 & 7)
Decommissioning

Fig. 1. Secure Development Lifecycle with Mapping to Practices of IEC 62443-4-1

This described high-level process can be further tailored and refined towards specific domains and system classes [2] and many aspects also correlate with security standards for specific domains. One being the industrial automation domain and its novel standard IEC 62443. This standard describes not only security requirements for industrial automation and control system components (e.g., PLCs) [4], but also requirements for their development process [3]. Regarding the product itself, so-called component requirements (CRs) and their enhancements (CR-REs) define countermeasures a product needs to implement. The higher the targeted security level (SL) of the product, the more CRs and CR-REs the product has to fulfill. This is depicted on the right of Fig. 2 (gray elements).

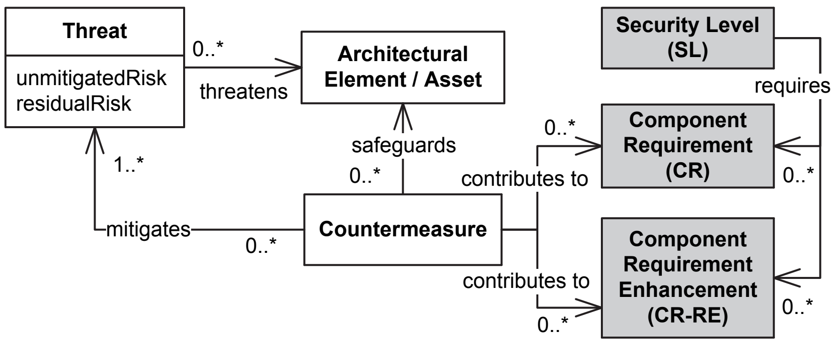

Fig. 2. Threat analysis elements for IEC 62443

Regarding the software development process, part 4-1 of the standard describes requirements for eight mandatory security practices that must be met to create IEC 62443 compliant industrial automation components [3]. In Fig. 1, we mapped the numeration of the required security practices in the standard (indicated in brackets) to the secure development lifecycle. In the following, we shortly describe the different security practices and their purpose to provide context. In this paper, however, we focus on the security practice threat analysis.

The very first security practice in the secure development lifecycle is *threat analysis* (corresponds to *Practice 2*). Threat analysis is used to identify the system's assets, its security objectives, threats to it, and associated risks. Using these results of the threat analysis a *secure design* can be created (corresponds to *Practice 3*), which specifically targets the identified threats with appropriate countermeasures on an architectural level. This ensures that the risk of the threats is reduced to an acceptable level and the product design is secure. The next security practice is *code analysis* (*Practice 4*). The realization of secure design can introduce new vulnerabilities, e.g., due to flaws in the code like buffer overflows. Thus, code analysis is applied to find and fix flaws possibly present in the code. Afterwards, *security tests* and *penetration tests* should be performed (*Practice 5*). Security tests validate that appropriate mitigations for each threat are in place as test cases are defined and executed for each identified threat. In contrast, the penetration test is used to exploratory test and ensure that there are no other possible ways to break into the developed product. When setting up products in the field, the process demands *hardening*, as well as continued *secure operation* (*Practice 8*). This security practice is important, as many vulnerabilities are caused by faulty deployments or arise through misconfiguration or reconfiguration during operation. The *update management* (*Practice 6 & 7*) is needed, as the product is usually used over an extended period of time. In this way, the process also covers the long-term security of the created product. Its purpose is to be able to quickly fix vulnerabilities after the product has been released in order to restore its security. As these updates need to be developed securely as well, they need to run through all the described security practices. Lastly, the process needs to encompass the product's decommissioning, e.g., by destroying sensitive data.

3 Threat Analysis Compliant with IEC 62443

To design a threat analysis practice that is compliant with IEC 62443 and customized to the organization of Phoenix Contact, we first had to analyze the current situation described in Sec. 3.1. Based on the given situation, we elicited requirements that our solution would have to fulfill. These are elaborated in Sec. 3.2. The following Sec. 3.3 explains the threat analysis solution we developed, and Sec. 3.4 argues the fulfillment of the requirements posed.

3.1 Initial Situation

Phoenix Contact had a well-established hybrid development process in place. Their overall V-model process covers the whole interdisciplinary development, starting from the product idea it covers requirements, development, testing, and finally the continued support. Within this overall V-model, the software development takes place between two set milestones, and, depending on the project, is performed agile or according to the V-model. In order to be able to manage such a process, Phoenix Contact uses the application lifecyle management (ALM) tool Polarion[4]. It is used to specify work items for requirements, design specifications, development tasks, test cases, and defects (cf. Fig. 3). A crucial role of Polarion is ensuring the traceability throughout the whole process. Each aforementioned work item is traced to its ancestor work item. Therefore, it is well documented which specification, task, or test case impacts which requirement.

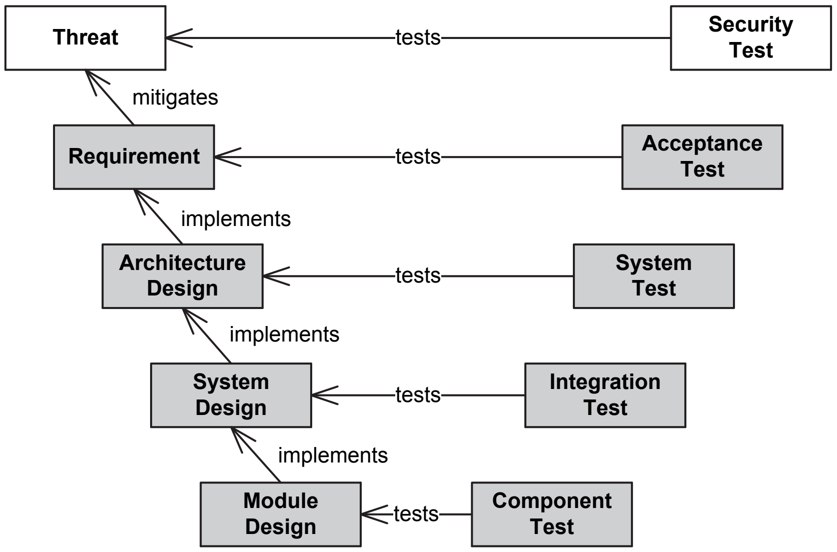

Fig. 3. V-model of Polarion work items extended for threat analysis (white boxes)

It was important that the designed threat analysis supports an iterative and repeated execution, so that it fits well with the existing V-model and agile process. In order to implement the threat analysis, it was necessary to adapt the process, but also the tools. The goal being that they are well integrated with Polarion, while fulfilling organizational and standard-driven requirements as listed in the following section.

[4] https://polarion.plm.automation.siemens.com/

3.2 Threat Analysis Requirements

A threat analysis practice consists of a process to follow and work products to create. The process for threat analysis that was integrated into the existing Phoenix Contact development process, had to meet requirements prescribed by the standard IEC 62443 [3, 4] and requirements posed by the organization.

The IEC 62443 poses the following requirements [3]:
(1) A threat model with the system's architectural elements, assets, security context, attack vectors, and trust boundaries shall be specified.
(2) Threats shall be identified, including the assessment of their unmitigated and residual risk.
(3) Countermeasures shall be defined that contribute to the CRs and CR-REs.
(4) The traceability of work products, from threats to countermeasure requirements and test cases shall be ensured.

Fig. 2 depicts the elements of these requirements and their relationship. The white elements are required for threat analysis and the gray elements to assure that the standard's product security requirements are met [4]. *Threats* threaten *architectural elements* or *assets* of the system under development. *Countermeasures* are implemented to safeguard the system and mitigate one or more threats. Threats constitute two types of risk: an *unmitigated risk* if no countermeasures are taken and a *residual risk* with countermeasures in place. The countermeasures shall not only mitigate the threats but shall also contribute to the *CRs* and *CR-REs* required by the targeted *security level*. Hence, for the threats that are identified in threat analysis, countermeasures have to be defined that cover the set of CRs and CR-REs. Doing so ensures that the countermeasures for the fulfillment of the CRs/CR-REs are justified by true threats identified in threat analysis.

In addition to the requirements prescribed by the standard, we also elicited requirements stemming from existing processes, tools, and people:
(5) The threat analysis process shall integrate into the existing hybrid development process and be applicable in the V-model and agile approach.
(6) The effort for threat analysis during product development shall be as low as possible (reasonable and manageable).
(7) The threat analysis process shall be accepted by all stakeholders.
(8) Hence, the stakeholders shall be supported by tools that guide the analysis and automate tedious, recurring, and error-prone tasks.
(9) These tools shall integrate with the existing development tool-chain.

3.3 Threat Analysis Solution

Based on the requirements listed in the previous section and Fraunhofer IEM's threat analysis approach [1], we designed a tool-supported threat analysis process and integrated it into the existing development process at Phoenix Contact. In that development process, work products are specified as Polarion work items as depicted in Fig. 3. The gray boxes represent existing work items and the white boxes represent work items that we added for the integration of threat analysis. In the existing process, requirements are implemented and refined by architecture, system, and module designs. In addition, tests are specified for each design level and the requirements. For threat analysis, we added the threat work item and according security test (the white boxes in Fig. 3). The threats are mitigated by requirement work items that describe security requirements for countermeasures to be implemented.

In the following, we describe each step of the threat analysis process (cf. Fig. 4). The Steps 1 to 4 are executed by a team consisting of product managers, product owners, software architects, developers, and security experts. Typically, each step is executed in form of a workshop with approx. five participants from the mentioned roles and moderated by a security expert. Step 5 represents the default development process and, hence, is executed by software architects and developers.

Step 1 In the first step, the system architecture and its security context is specified in a *threat model*. The system architecture is specified as a data flow diagram that contains the components that interact with the system's environment, their assets, their interaction path, and the used communication technologies. In addition, trust boundaries are specified to identify different zones of trust. Together with the system architecture, the system's security context is defined. It specifies the assumptions about security measures in the environment of the system, e.g., precautions taken by the customer.

The threat model is specified using the Microsoft Threat Modeling Tool (TMT) which is available free of charge[5]. It supports threat analysis by providing a so-called *threat model template* that defines the usable types of model elements. For compliance with IEC 62443, we tailored the template to the industrial automation domain to account for the domain-specific taxonomy, technologies, and threats. By using TMT to document the information described above, the threat model contains all details required by IEC 62443 (cf. Requirement 1 in Sec. 3.2).

Step 2 In this step, threats to the system specified in Step 1 are identified using the STRIDE approach [9] and analyzed. For each identified threat, the risk of exploitation is assessed, if no countermeasures are taken (i.e., the unmitigated risk).

Based on the specified threat model and the threat model template, TMT generates threat candidates for each STRIDE category. The team has to analyze those threat candidates for applicability. In addition to the generated threat candidates, further threats have to be identified by creative thinking.

Valid threats to the system are transferred from TMT to Polarion threat work items using a tool that we developed specifically for that purpose (cf. *threat synchronization by custom tool* in Fig. 4).

[5] https://aka.ms/tmt

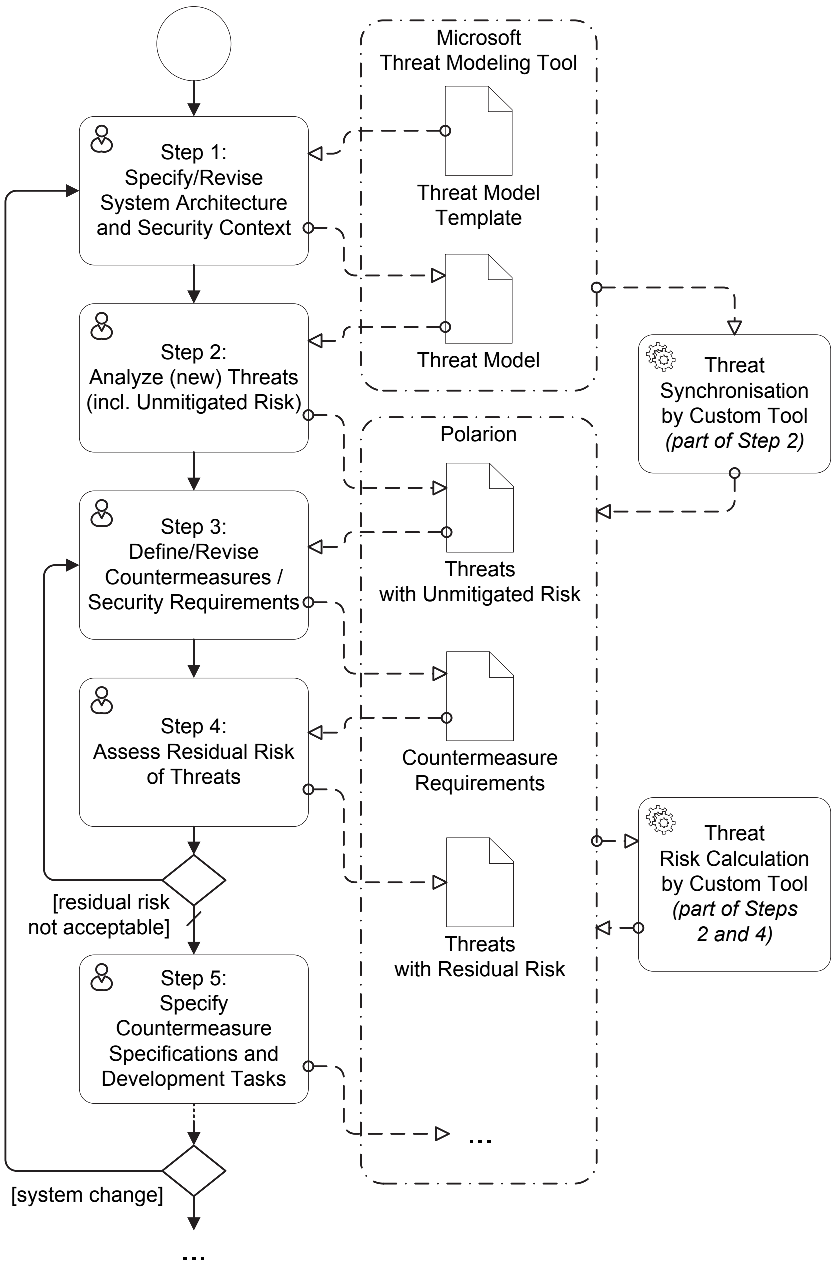

Fig. 4. Threat analysis process

For risk assessment, the team determines threat severity using the Microsoft Bug Bar[6] that was adapted by us for the use with PLCs. Our Bug Bar has five severity levels (from negligible to catastrophic). Separately for each STRIDE category, it defines what consequences map to what level (concerning just the PLC, not the system it may be used in). E.g., a tampering threat is considered catastrophic if it gives the attacker full control of the PLC. This allows the team to easily read off the standardized severity for every STRIDE threat in the context of PLCs. In addition, they determine threat likelihood by combining four additional metrics derived from the IEC 62443 security level definitions and CVSS[7]. These are required skills, required resources, required privileges, and attack vector. Based on the determined metric values, a tool that we developed, automatically calculates the risk value and sets the according property of the respective threat work item (cf. *risk calculation by custom tool* in Fig. 4).

The result of Step 2 is a list of threats with their unmitigated risk as required by IEC 62443 (cf. Req. 2 in Sec. 3.2).

Step 3 In the third step, countermeasures are defined that mitigate the threats identified in Step 2. The countermeasures are specified as requirement work items in Polarion. For traceability, each countermeasure requirement is linked to the threats it mitigates (cf. Fig. 3) and the IEC 62443 CRs and CR-REs that it contributes to (cf. Fig. 2). In addition, these requirements are linked to architecture design specifications and acceptance test cases as defined in the existing Phoenix Contact development process (cf. Fig. 3). This fulfills Req. 3 and 4 from Sec. 3.2.

Step 4 In this step, the risk of each threat is determined that remains after the defined countermeasures are in place (i.e., the residual risk). This risk is determined using the same assessment approach as for the unmitigated risk in Step 2, i.e., using the adapted bug bar and likelihood metrics and the custom risk calculation tool. If the residual risk is considered too high (not acceptable), additional countermeasures have to be defined by returning to Step 3. The result of Step 4 is a list of threats with their residual risk as required by IEC 62443 (cf. Req. 2 in Sec. 3.2).

After Step 4, the results are a list of threats with unmitigated and residual risk (specified as threat work items in Polarion) and a list of countermeasures mitigating those threats and contributing to the standard's CRs/CR-REs (specified as requirement work items in Polarion).

Step 5 ff. Once all countermeasure requirements are specified and all residual risk is accepted, the countermeasures have to be designed and implemented in accordance with the existing development process. This is represented by Step 5 in Fig. 4. The design of countermeasures is specified (cf. architecture, system, and

[6] https://docs.microsoft.com/en-us/security/sdl/security-bug-bar-sample
[7] https://www.first.org/cvss

module design in Fig. 3) and each module design is assigned with development tasks for implementation.

After Step 5, whenever the system is changed such that the threat model is no longer in sync with the development state (e.g., at start of a new sprint), the threat analysis process is repeated from Step 1. The threat model is updated to reflect the system changes, newly introduced threats are analyzed, countermeasure requirements are revised/added, and the risks are updated.

3.4 Requirements Fulfillment

The fulfillment of the four requirements posed by IEC 62443 as listed in Sec. 3.2 is described in the previous section. The five additional requirements posed by Phoenix Contact are fulfilled by the following measures:

To integrate threat analysis into the existing development process (cf. Req. 5 in Sec. 3.2), we made sure it can be iteratively refined (e.g., in following sprints) and uses established work items (cf. Fig. 3). For instance, countermeasure requirements are specified in the same way as any other requirement.

To keep the effort for threat analysis on a reasonable level (cf. Req. 6), we support the engineers by tooling that is tailored to the domain, automates tedious tasks, and integrates with the existing tool Polarion that the stakeholders are used to.

To guide threat analysis (cf. Req. 8), we use TMT and a tailored threat modeling template. The template's diagramming palette helps to specify the system architecture, and the threat generation helps to identify typical threats relevant to the domain. Additionally, to avoid tedious, recurring, and error-prone tasks, the two developed tools automatically import threats to Polarion and calculate the unmitigated and residual risk of threats in Polarion (cf. Req. 8 and 9).

Finally, to make sure the threat analysis process is accepted by all stakeholders (cf. Req. 7), we scheduled repeated workshops with them to describe the state of the process and its tools and to get feedback for changes. The tool-support to guide the analysis, automate tedious tasks, and integrate with the existing Polarion also played a key role for stakeholder acceptance.

4 Lessons Learned

During the design and integration of an IEC 62443 compliant threat analysis in the existing development process, we made several key findings which we present in the following. The findings can be categorized in findings concerning the process and its usage, as well as concerning the threat analysis itself.

4.1 Findings Concerning the Process

The integration of threat analysis into an existing hybrid development process (V-model and agile) is achievable. We found that it is very important to integrate the new threat analysis process well into the existing processes by having proper tool support. We see this as a confirmation of the principles of the SPI manifesto. Especially the tool support helps all stakeholders to easily understand the purpose of threat analysis and how to execute it. Furthermore, the integration into Polarion, the tool which every stakeholder is already using on a daily basis, is a vital element for the adoption of the new process, as they are already used to Polarion and are not forced to constantly switch between tools. One first benefit was also already achieved by solely constructing a diagram of the system architecture and its trust boundaries (Step 1 in Sec. 3.3), even without generating threats, as it already serves as a good visualization for threat discussions among stakeholders.

4.2 Findings Concerning the Threat Analysis

We found that tailoring the used threat template of the Microsoft Threat Modeling Tool to the specific domain is crucial for the usefulness of the generated threats (architectural elements, threat candidates, avoiding redundant and non-applicable threats). In addition to the generated threats, it is still important to creatively think about possible further threats to the system, for which the discussion in a group proved helpful. Concerning the CRs defined by IEC 62443, we found that they differ in their abstraction level and also cannot be seen as countermeasures that can be directly mapped to threats. That is, one CR can be a countermeasure to several threats, but also the other way round, one threat may require several countermeasures.

4.3 Further Findings

In order to prioritize the found threats properly, risk assessment is necessary. However, assessing risk is a very difficult task for industrial automation component manufacturers like Phoenix Contact. The reason is that determining the severity of an exploit (e.g., the resulting damage) is highly dependent on the context in which the industrial automation component is used. The context also cannot be unified, as industrial automation components, such as PLCs, are used in a wide field of application with different criticality (e.g., power plants or small product lines). The usage and adaptation of the Microsoft Bug Bar proved to be a successful way to circumvent this problem. It was used to apply severity assessment of a threat solely by the threat's STRIDE category in context of the product, i.e., the field of application is not required.

According to the values of the SPI manifesto, the organizational change needs to be well managed. This is also what we found, as the introduction of the threat analysis into an existing process required in our case approximately one and half years, which would have been impossible without proper organization.

5 Conclusion

In this paper, we used threat analysis as an example to illustrate, how we designed IEC 62443 compliant security practices and implemented them into the existing development processes for PLCs at Phoenix Contact. Phoenix Contact has been successfully certified for compliance with IEC 62443 by TÜV SÜD AG. We explained that beside product requirements, the standard also describes process requirements for a secure development lifecycle of industrial automation and control systems. Tailoring existing development processes to fulfill these requirements is currently a major hurdle for organizations as concrete best practice implementations are still missing for the novel standard. The required security practices cover the whole development lifecycle and we focus on threat analysis in this paper, which is the very first. We discussed the requirements for threat analysis posed by the standard and also explained that in addition, Phoenix Contact requires a threat analysis with little friction that is accepted by stakeholders and well-integrated into the existing process and tool-chain. We then presented the implemented solution that makes use of the STRIDE approach and the Microsoft Threat Modeling Tool with an integration to the company-standard application lifecycle management solution Polarion. We learned that investing into customization and automation as well as into integration with the existing tool chain were key success factors for the active and effective adoption of the altered process in terms of the SPI manifesto. In the future, we will adapt the threat analysis for the development of other components by Phoenix Contact. In addition, we work on the enhancement of risk management on system level based on the illustrated component-level threat analysis. Fraunhofer IEM is constantly improving and customizing its threat analysis approach [1] for further domains and companies.

References

1. Fockel, M., Merschjohann, S., Fazal-Baqaie, M.: Threat analysis in practice – systematically deriving security requirements. In: 19th Int. Conf. on Product-Focused Software Process Improvement (PROFES 2018). Springer International (2018)
2. Geismann, J., Gerking, C., Bodden, E.: Towards ensuring security by design in cyber-physical systems engineering processes. In: International Conference on Software and System Process (ICSSP 2018) (May 2018)
3. IEC: IEC 62443-4-1:2018: Security for industrial automation and control systems – Part 4-1: Secure product development lifecycle requirements (Jan 2018)
4. IEC: IEC 62443-4-2:2019: Security for industrial automation and control systems – Part 4-2: Technical security requirements for IACS compnents (Feb 2019)
5. Macher, G., Armengaud, E., Brenner, E., Kreiner, C.: A review of threat analysis and risk assessment methods in the automotive context. In: Computer Safety, Reliability, and Security. Springer International Publishing, Cham (2016)
6. Macher, G., Messnarz, R., Armengaud, E., Riel, A., Brenner, E., Kreiner, C.: Integrated safety and security development in the automotive domain. In: SAE Technical Paper. SAE International (03 2017)
7. Network, E.U.A.F., Security, I.: ENISA threat landscape report (2016)
8. Pries-Heje, J., Johansen, J.: SPI manifesto (2010), version A.1.2.2010
9. Shostack, A.: Threat Modeling: Designing for Security. Wiley (Feb 2014)
10. Traub, M., Vögel, H., Sax, E., Streichert, T., Härri, J.: Digitalization in automotive and industrial systems. In: 2018 Design, Automation Test in Europe Conference Exhibition (DATE). pp. 1203–1204 (March 2018)