Towards Computer-Aided Security Life Cycle Management for Critical Industrial Control Systems

Florian Patzer[1(✉)], Ankush Meshram[2], Pascal Birnstill[1], Christian Haas[1], and Jürgen Beyerer[1,2]

[1] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB), Karlsruhe, Germany
{florian.patzer,pascal.birnstill,christian.haas,juergen.beyerer}@iosb.fraunhofer.de

[2] Vision and Fusion Laboratory (IES), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{ankush.meshram,juergen.beyerer}@kit.edu

Abstract. Critical infrastructure experienced a transformation from isolated towards highly (inter-)connected systems. This development introduced a variety of new cyber threats, causing high financial damage, threatening lives and affecting the society. Known examples are Stuxnet, WannaCry and the attacks on the Ukrainian power grid. To prevent such attacks, it is indispensable to properly design, assess and maintain countermeasures and security strategies throughout the whole life cycle of the critical systems. For this, security has to be considered and assessed for every system design and redesign. However, common assessment tools and methodologies are not executed on a detailed system knowledge and therefore they are enhanced with penetration tests. Unfortunately, performing only abstract assessments is inadequate and penetration tests endanger the availability of the tested systems. Therefore, the latter cannot be performed on live systems executing critical processes. In this paper, we address these issues for Industrial Control Systems and explain how new concepts for continuous security-by-design or model-based system monitoring and automated vulnerability assessments can resolve them by exploiting new Industry 4.0 developments.

Keywords: ICS security · Critical infrastructure security
Security-by-design · Automated vulnerability assessment
Security life cycle management · Defense-in-depth · Knowledge base

1 Introduction

Many industrial systems (hereafter referenced as Industrial Control Systems (ICS)) are classified as critical infrastructure. Due to high costs of down-times or the respective risk for safety and public health, interruption of their processes

E. Luiijf et al. (Eds.): CRITIS 2018, LNCS 11260, pp. 45–56, 2019.
https://doi.org/10.1007/978-3-030-05849-4_4

is usually unacceptable. Consequently, in contrast to office IT, the prioritized security objective for ICS is availability and not confidentiality.

Unfortunately, ICS are generally very vulnerable to cyber attacks. One reason for this situation is that the applied technologies were not designed to fulfill security requirements, since the systems in question were isolated from the outside world for decades and thus isolated from many kinds of attacks. As a consequence, the need for security has not been strong enough to support the development of more secure technologies. In addition, the life time of ICS components tends to be much longer than that of components in other domains. Thus, insecure technology is still common in ICS. However, the mentioned isolation does not exist anymore and most ICS operators have realized that to maintain availability and safety of their systems, they have to apply countermeasures which will prevent attackers from endangering the systems.

Additionally, governments have reacted to the new threats by submitting new laws which try to force the ICS operators to improve their systems' security. For example, by building security management systems and performing respective security audits. To support the ICS operators at the design and implementation of these measures, standard collections such as IEC 62443[1] and NIST SP 800-82 - Guide to Industrial Control Systems (*ICS*) Security[2] have been elaborated.

Among others, the standards contain measures to maintain the security management systems and to ensure their effectiveness. This includes periodic vulnerability assessments, which are typically performed via "pen and paper" approach and supplemented with penetration tests. However, pen and paper assessments often cannot rely on a detailed technological view of the system (1). The available analyzing and testing techniques applied in penetration tests on the other hand can lead to malfunctioning and outage of the systems under test (2). To avoid such disruptions, in ICS penetration tests can be either omitted or just performed on isolated test systems. The former is very dangerous, since there is not even an indication for the effectiveness of the applied security measures, for existing vulnerabilities or for system design-flaws. Thus, such issues are often first recognized, once an attacker has already exploited them. To avoid this, the common solution is the analysis and penetration of isolated testbeds [3]. Nevertheless, we argue that due to differences in configuration, state and in- or outbound interfaces of the perimeter the test systems are not identical to the real systems. Thus, this approach is generally inaccurate (3). In modern flexible plants, such rather static testbeds would even be incomparable to the real system.

Moreover, the security of a system has to be assessed and improved over its whole life cycle, meaning planning, engineering, deployment, operation, maintenance, adaptation and decommissioning of the system and its components. In terms of security-by-design this includes not only every system design but also every redesign. Such ongoing security evaluations and assessments require current knowledge about the evolving system and do not scale without proper

[1] https://isa99.isa.org/ISA99%20Wiki/Home.aspx.
[2] https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final.

computer-aided security analysis. To the best of our knowledge, solutions supporting these necessary processes are currently not available for ICS (4).

Leveraging new Industry 4.0 concepts, such as digital twins (cf. Sect. 3) and interoperable data exchange protocols for ICS devices, new and enhanced security applications can be realized. These applications operate on detailed technological information about the respective critical system without stressing the system's components and networks while providing comprehensive security analysis and security life cycle management. We argue that with such security applications the above mentioned issues (1)–(4) can be resolved.

In this work-in-progress paper, we introduce first concepts and results of our research regarding such security applications (cf. Sects. 2.1, 2.2, 2.3 and 2.4). All these applications rely on a computer-readable system knowledge base (i.e. a digital twin). Moreover, we describe what kind of information is needed to be collected by the knowledge base and how it is collected (cf. Sect. 2). Afterwards, we discuss similar work and applicable related Industry 4.0 concepts (cf. Sect. 3). Finally, we provide a discussion about specific issues and pitfalls which arise when such a knowledge base is implemented for the ICS domain (cf. Sect. 4).

2 System Knowledge Base

Each device of an ICS comprises different types of security-relevant information. Such information can be a device's network configuration, software details, applied security measures or available hardware interfaces. These classes can further consist of subclasses. For example, a network configuration can contain static information (e.g. a MAC address) and dynamic information, e.g open ports. Open ports relate to services which can be described with static information, like the type of a service and its version. In our concept, we let each device (further called *Device of Interest* (*DoI*)) of the ICS hold a semantically described model containing this information. Periodically or when certain values have changed, the device updates its model's corresponding values and offers this model to the knowledge base (see Fig. 1). We call this process *self-disclosure*. By building the knowledge base from this information, a digital copy of the real system is generated and maintained. The Platform Industry 4.0 consortium[3] is currently developing the Asset Administration Shell [12] which will implement such a self-disclosure strategy (cf. Sect. 3) and should ideally be implemented on every future component.

A second source of information are models sent by a user (user-based information). As an example, this user could be an engineer creating a system design or redesign using engineering tools. As output, these tools can generate a semantic representation of the designs, e.g. as AutomationML model [15] or OPC UA node set [1,7] which then embodies a model that can be taken as input by the knowledge base.

Using these two types of information, the knowledge base consists of a representation of the real system and another of the system's design. These two

[3] https://www.plattform-i40.de/I40/Navigation/EN/Home/home.html.

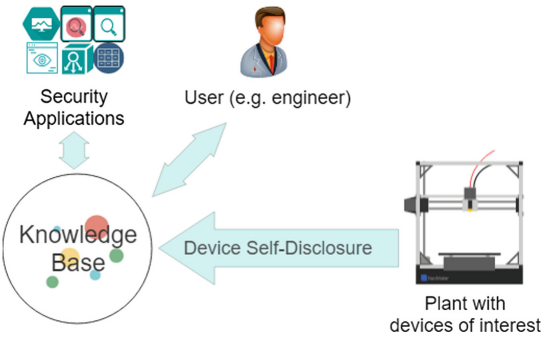

Fig. 1. A system knowledge base interacting with different agents

representation types can now be used by different security applications. Certainly, such a knowledge base can also be used by other types of applications, but in this publication we concentrate on the security domain.

In the following sections, we describe the security applications we deem most important to provide a high level of security in ICS. All these applications exploit the knowledge base.

2.1 System Flaws and Continuous Security-by-Design

The first new security application is a vulnerability analysis tool for a type of vulnerability we classify as *system flaw*. This class consists of vulnerabilities which are not found in common vulnerability databases and are not zero-day exploits. Instead, system flaws are flaws within an ICS that can have a negative impact on its security, for example, missing security measures like firewalls, or a flawed network segmentation due to wrong VLAN affiliations, or undesired conduits through dual homed computers. Further examples are policy violations within a security zone due to firewall misconfiguration or hardware interfaces that are not allowed, but nevertheless available. A *System Flaw Analyzer* (cf. Fig. 2) can look for such a vulnerability by retrieving relevant information from the knowledge base. It can then transform this information into facts (knowledge facts) (e.g. Prolog facts[4]). The vulnerability being searched also gets described as facts (vulnerability facts). The analyzer can use these facts as input for a reasoner (e.g. Prolog) which will calculate whether all these facts can be true simultaneously. As a simplified example, the knowledge fact $is\_open(device, port)$ and the vulnerability fact $not(is\_open(device, port))$ cannot be true at once. Thus, the System Flaw Analyzer can find these issues. The vulnerability facts can consist of facts describing attack vectors, which formally describe what conditions the attack would exploit, and facts representing security policies, describing what conditions are allowed (or not allowed). This idea is mainly motivated by the

[4] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html#facts.

vulnerability analysis of MulVal ([11], cf. Sect. 3) which uses this approach but neither semantic models nor ICS applicable data exchange protocols.

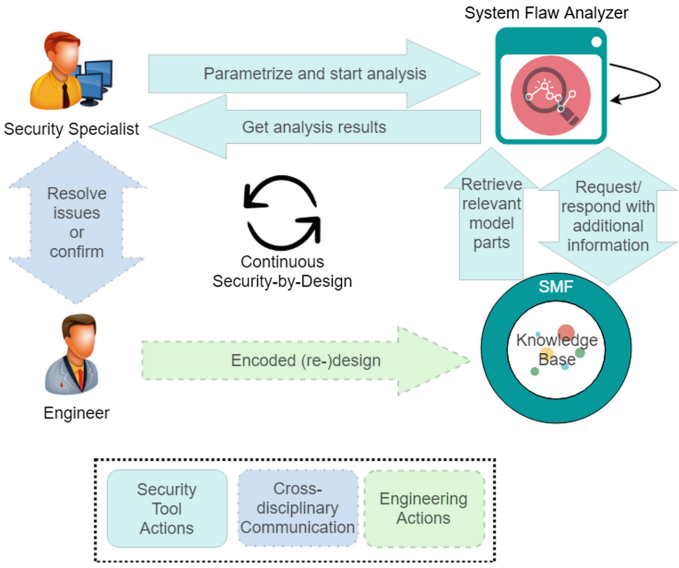

Fig. 2. The concept of continuous security-by-design

Until now the System Flaw Analyzer has only used the ICS representation. However, by adding another step of intelligence to the analyzer, it can be used for a concept we call *continuous security-by-design*, which can be seen in Fig. 2. For this, the aforementioned engineer submits the model of a partial redesign to the knowledge base. As described before, this can be created using common engineering tools. Let's assume the redesign model consists of a certain device reconfiguration, namely a new IP address (this is a very simplified example for better understanding). The System Flaw Analyzer can then be configured by a security specialist to use the main model and apply the redesign to it. Subsequently, the analyzer proceeds as described before but with one exception. Every time it requests information from the knowledge base's ICS representation to build the knowledge facts, it replaces the retrieved information with the corresponding information of the redesign. In our example, it would replace the device's current IP address with the one specified in the redesign. As a result, the analyzer would calculate on facts, representing the system as it will be when the redesign is applied. The security specialist can then use the analyzer's output to identify necessary changes and measures. He can then assist the engineer to improve the security of the redesign, which afterwards can be resubmitted to the knowledge base. This process can then be repeated until the security specialist has no more concerns about the changes. This allows a detailed computer-aided assessment of the real ICS **before** it gets adapted. Thus, in contrast to common

automated vulnerability assessment concepts, our approach is able to find security vulnerabilities before they exist in the real system and does not endanger the system at any point in time.

Moreover, if the analyzed system model has the granularity of device configurations, it can even be used to (re-)configure the devices of interest accordingly, as it is already common for network devices via NETCONF [5].

2.2 Known Vulnerabilities

Another type of vulnerability can also be identified using the knowledge base, namely the *known vulnerabilities*. These vulnerabilities can be found in public databases like the Open Source Vulnerability Database (OSVDB[5]). The already referenced application MulVal [11] uses such databases. It sends the vulnerability precondition descriptions, retrieved from such databases, to the devices which use special scanners to compare them to their own configuration and state. If this comparison results in a successful match, the device notifies the MulVal main application about the match, which can then conclude that the device has the respective vulnerability.

Such applications can minimize the reaction time to new exploits and help to manage them, e.g. by keeping track of them until patches are available. Thus, it is desirable to support such applications for ICS. This goal can be achieved, when the application (e.g. MulVal) queries the knowledge base for devices matching the preconditions, instead of asking every device to perform a self-check for the preconditions. The result would contain the vulnerable devices.

2.3 System Model Monitoring

In every complex ICS, the original design and the actual system tend to diverge from each other as the time proceeds. In modern plants, this is even intended since more and more flexibility within the system composition is introduced, e.g. by Plug-and-Produce concepts [4]. Especially from a security point of view this development is dangerous, since security measures cannot be maintained accurately if the available view of the system is outdated. For example, firewall rules might not be updated correctly, when the removal or addition of a device which communicates through the firewall is not recognized.

However, when the knowledge base contains the representation of the design and the real system, a security application, we simply call *System Model Monitoring*, can compare these models to each other (e.g. on every update of the knowledge base) and raise alerts when they differ. This enables the engineer and security specialist to timely react to the issue.

2.4 Testbed Synchronization

We already argued why testbeds are important for ICS. Furthermore, as mentioned before their main disadvantage is their disability to represent the real

[5] https://blog.osvdb.org.

system properly. The knowledge base consists of the information necessary to synchronize a testbed with the real system. If the testbed devices support the necessary protocols (cf. NETCONF Sect. 2.1), their configuration can be updated by the knowledge base, or an additional synchronization application. As a result, tests can be performed on a testbed, which is more comparable to the real system.

There are even more security-related applications being out of scope for this paper. Examples are the context provision for intrusion detection systems, the improvements of pen and paper security assessments or the retrieval of crystal box knowledge when penetration tests are being applied after all.

2.5 Modeling Language

The two types of models, self-disclosure- and user-based, are the core of the knowledge base. Nevertheless, we did not define the modeling language which is required to create and encode semantic models. A number of languages are available to build such a model. Unfortunately, the available modeling languages either support some of the necessary semantics but cannot be applied for reasoning (i.e. they are not directly convertible into facts, cf. Sect. 2.1) or they do not support the necessary semantics but do support reasoning. However, various available ICS devices already support protocols for information disclosure, focusing on interoperability and therefore providing own modeling strategies. As an example, industrial devices often support either OPC UA or oneM2M[6], whereas for network devices NETCONF with YANG [2] is more common. Since the resource restrictions of the DoIs will usually inhibit the simultaneous existence of multiple such protocol stacks and it would be futile to expect vendors of different domains to deploy a common protocol stack only used for the here described concepts, the knowledge base should be able to communicate with the DoIs using various protocols. Thus, a transformation from different languages into a common one, which is also supported by reasoners, is inevitable. For example, our current implementation of the knowledge base is connected to the DoI's via OPC UA (cf. Fig. 3) using the protocol's own information model. The received data gets then converted into OWL 2 DL[7] to be able to perform reasoning and leverage the strong tool support for OWL 2 DL. Nevertheless, the here described concept of the knowledge base is not restricted to any certain modeling language.

3 Related Work

Critical processes within ICS will be more and more equipped with so called *digital twins* [14]. Digital twins are digital representations of their physical "twin system". They do not have to consist of simulations or visual representations.

[6] http://www.onem2m.org/.
[7] https://www.w3.org/TR/owl2-primer/.

```
<UAVariable BrowseName="1:Destination" DataType="String" NodeId="i=11320" ParentNodeId="i=11316">
  <DisplayName>Destination</DisplayName>
  <Description>Destination</Description>
  <References>
    <Reference ReferenceType="HasModellingRule">i=78</Reference>
    <Reference IsForward="false" ReferenceType="HasComponent">i=11316</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
  </References>
  <Value>
    <uax:String>anywhere</uax:String>
  </Value>
</UAVariable>
<UAVariable BrowseName="1:Source" DataType="String" NodeId="i=11321" ParentNodeId="i=11316">
  <DisplayName>Source</DisplayName>
  <Description>Source</Description>
  <References>
    <Reference ReferenceType="HasModellingRule">i=78</Reference>
    <Reference IsForward="false" ReferenceType="HasComponent">i=11316</Reference>
    <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
  </References>
  <Value>
    <uax:String>anywhere</uax:String>
  </Value>
</UAVariable>
```

Fig. 3. Example snipped of an OPC UA-based self-disclosure as XML export which shows the source and destination fields of an IP Tables entry.

Instead a digital twin can consist of only a semantic model of a system. Depending on the use case, it can be composed of data with different focus. Common digital twin concepts often focus on physical or process-related data. Based on this data, applications like simulations, analysis or monitoring can be performed without having to alter, endanger or stress the real system. The knowledge base described in Sect. 2 is therefore a digital twin of critical systems, concentrating on security-relevant information.

Although network components already support self-disclosure for years (e.g. NETCONF or OF-config for Software-Defined Networking [10]), PLCs, human machine interfaces (HMIs), industrial PCs or smart sensors either do not support similar concepts or only for process-related data. However, the trend towards such technology is already visible. For example, flexible and self-orchestrated production concepts like the Asset Administration Shell (AAS) [12] and concrete protocols like OPC UA have been developed. The AAS is a concept designed precisely to empower devices to provide the service of self-disclosure. It even consists of a security view, which should provide security-related information as desired for security assessments. The amount of important industrial partners of the AAS project shows the need for such a technology and strengthens the impression that a wide range of future industrial components will support this technology. Unfortunately, the concept and its implementations are not yet sufficiently mature to be applicable for our approach. In the future, AAS implementations might provide a suitable self-disclosure solution for the here described concept (cf. Sect. 2).

Automated vulnerability assessment tools, like OpenVAS[8], have been available for years. However, most of these tools perform the same scans and penetration techniques as manual penetration tests. As argued before, these are not

[8] http://www.openvas.org/index.de.html.

applicable for critical systems and are therefore not further considered. More relevant approaches are explained in the next paragraphs.

The Open Vulnerability Assessment Language (OVAL[9]) is a language to encode configuration and vulnerability details for vulnerability assessments. An OVAL scanner running on the device under test, can perform automated vulnerability assessments by receiving OVAL-encoded vulnerability and related configuration descriptions from a remote OVAL main application and comparing them to its system's configuration. However, due to their local view of the device, OVAL scanners are limited to device vulnerabilities. Thus, they are not applicable for the System Flaw Analyzer concept of Sect. 2.1. Even though to the best of our knowledge no scanners and schemes exist, which can be applied to common industrial components, it might, for example, be reasonable to support OVAL as a language to interact with the knowledge base and use the OVAL Systems Characteristics Schema to describe device configurations (cf. Sect. 2.5).

A system analysis approach using OVAL scanners is MulVal [11]. MulVal is a concept which gathers vulnerabilities of devices by using OVAL scanners. Additionally, network information is captured via routers and firewalls. All this information is sent to a host running the main application. This application transforms the information into Datalog, which is a subset of Prolog and can therefore be transformed into Prolog facts directly. The same main application receives a list of rules, written in Datalog as well, which define semantics of different kinds of exploits, compromise propagation, whitelist access policies and multihop network access. As a result, Prolog can be run as a reasoner given the facts derived from the device/network information and the facts representing the rules. The idea of letting the devices provide security relevant information to a system by themselves and running reasoners on that information is similar to our approach in Sect. 2.1. However, MulVal is not designed for ICS, collects facts instead of building semantic models and the reasoning concentrates on attack graphs given the device vulnerabilities instead of analyzing the facts for compliance to best practice. In contrast to the MulVal assumptions, in the industrial domain various data formats and protocols have to be supported for information gathering, and simultaneously a variety of different applications will use this knowledge base (e.g. our security applications, digital twin implementations or inventory tools). Even though in [13] the authors claim to propose a MulVal-related solution for ICS, we could not find any evidence for that in the paper, since the solution does not consider any of the typical ICS devices, architectures or operating systems.

Further publications are available, either similar to MulVal [17], or extending it to perform risk assessments using game theory [8], or focusing on the model refinement [16]. The latter is an approach similar to our Unknown Vulnerability Analyzer concept. It conceptually leverages automated scanning, which is not recommended for ICS, as already mentioned, but can be used for our knowledge base approach as well.

[9] https://oval.mitre.org/language/.

A similar approach to MulVal was recently initiated in a series of IETF drafts and RFCs by the IETF Security Automation and Continuous Monitoring (SACM) working group[10]. The SACM WG describes a basic concept which consists of an OVAL-scanner-like as well as a self-disclosure approach. Currently, their work concentrates on endpoint security and the respective knowledge base is a software inventory. In addition, they do not yet consider reasoning on the captured data and do not take the peculiarities of the ICS domain into account. However, the working group is still active and intends to elaborate and specify further parts of the overall idea. Thus, in the future they might address more issues which could support the implementation and adaptation of our concept.

Model-based vulnerability assessments like [6] and [9] try to benefit from abstract models of the system (e.g. modelled in SysML[11]) in order to run automated vulnerability assessments. These approaches are not using real configurations and system states, which forces them to operate on a higher abstraction layer than our approach. Due to this abstract view, most vulnerabilities that could be identified on the knowledge base are not visible at the high level of SysML models or similar models. In other words, approaches based on such models have the issue of not representing the real system, but only an abstract plan or view of the system with no guarantees of validity. However, the strategies to identify vulnerabilities on high-level models might be useful to develop respective strategies for our concept. Moreover, it might be reasonable to use such abstract approaches in early design phases to support the planning of initial security measures and to use our approach afterwards.

4 Discussion

Since the knowledge base captures security relevant information about the DoIs, it can make the ICS even more vulnerable by providing an additional sweet spot for attackers. Hence, each communication with the knowledge base has to ensure confidentiality, integrity and authenticity of the data providing this information. Furthermore, we strongly advice that every knowledge base instance supports at least a level of security which is as high as the highest level of security provided by any of its DoIs. In addition, the same security level needs to be provided by the applications using the knowledge base and their respective environment. Moreover, the knowledge base should be hardened and tested intensively to ensure that no data can leak.

Most embedded devices currently do not support security algorithms such as en-/decryption, or signing and verifying messages. Therefore, they do not have the resources and dependencies (e.g. libraries) such mechanisms would require. Thus, our concept suffers from the same issue as the AAS, namely the infeasibility to deploy the necessary security on such devices. This sets a limit for the self-disclosure and respectively the knowledge base. To alleviate this issue, such

[10] https://tools.ietf.org/wg/sacm/.
[11] https://sysml.org.

devices can be reported by neighbored DoIs. Afterwards, additional information can be added by users manually to include the devices.

While developing applications using the data captured from self-disclosure, it is important to keep in mind that this data is only as trustworthy as the DoIs themselves. For some information e.g. regarding network interfaces this is actually a disadvantage of the here described self-disclosure approach which common network mapping techniques do not suffer from. However, for most information this issue remains regardless of what information gathering technique is applied.

5 Future Work

In this paper, we presented our concepts for ICS security life cycle management. For these concepts we currently implement a system knowledge base following the described approach of Sect. 2. As already mentioned, the current version of our implementation already supports OPC UA and a partial transformation to OWL 2 DL. Additionally, we are working on a first version of the System Flaw Analyzer for continuous security-by-design. This includes the evaluation of several reasoning and ontology based analysis techniques and a rich transformation from security best practices into logical statements. These implementations will help us to evaluate and improve our here explained concepts further.

6 Conclusion

In this paper we motivated the advantages of a new system knowledge base concept and how it improves and enables security applications. These security applications accomplish a level of security for critical systems which is currently not achievable. Unfortunately, currently available solutions of other domains are not applicable for critical ICS due to specific requirements, such as to attain technological compatibility to industrial protocols and modeling languages, security of the handled information, and feasibility of the knowledge base integration. Thus, we expect our current work to be essential for the future of ICS security.

References

1. PLCopen and OPC Foundation: OPC UA Information Model for IEC 61131–3. Standard, OPC Foundation, March 2010
2. Bjorklund, M.: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, RFC Editor, October 2010. https://rfc-editor.org/rfc/rfc6020.txt
3. CPNI: Cyber security assessments of industrial control systems: A good practice guide, April 2011
4. Dürkop, L., Imtiaz, J., Trsek, H., Wisniewski, L., Jasperneite, J.: Using OPC-UA for the auto configuration of real-time ethernet systems. In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN), pp. 248–253, July 2013. https://doi.org/10.1109/INDIN.2013.6622890

5. Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A.: Network Configuration Protocol (NETCONF). RFC 6241, RFC Editor, June 2011. https://tools.ietf.org/html/rfc6241
6. Holm, H., Sommestadt, T., Ekstedt, M., Nordström, L.: Cysemol: Atool for cyber security analysis of enterprises. In: 22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013), p. 1109. IEEE, Piscataway (2013). https://doi.org/10.1049/cp.2013.1077
7. OPC Unified Architecture - Part 1: Overview and Concepts. Standard, International Electrotechnical Commission, November 2016
8. Ji, Y., Wen, D., Wang, H., Xia, C.: A logic-based approach to network security risk assessment. In: 2009 ISECS International Colloquium on Computing, Communication, Control, and Management, pp. 9–14. IEEE, September 2009. https://doi.org/10.1109/CCCM.2009.5267887
9. Lemaire, L., Vossaert, J., Jansen, J., Naessens, V.: Extracting vulnerabilities in industrial control systems using a knowledge-based system. In: 3rd International Symposium for ICS & SCADA Cyber Security Research 2015. Electronic Workshops in Computing, BCS Learning & Development Ltd (2015). https://doi.org/10.14236/ewic/ICS2015.1
10. ONF: Of-config 1.2 - openflow management and configuration protocol - onf ts-016. Tech. rep., Open Networking Foundation (2014). https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf
11. Ou, X., Govindavajhala, S., Appel, A.W.: Mulval: a logic-based network security analyzer. In: Proceedings of the 14th Conference on USENIX Security Symposium, vol. 14. USENIX Association, Berkeley, CA, USA (2005). http://dl.acm.org/citation.cfm?id=1251398.1251406
12. Plattform Industrie 4.0: Structure of the administration shell, April 2016. https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/structure-of-the-administration-shell.pdf?__blob=publicationFile&v=7
13. Rakshit, A., Ou, X.: A host-based security assessment architecture for industrial control systems. In: 2nd International Symposium on Resilient Control Systems, pp. 13–18. IEEE (2009). https://doi.org/10.1109/ISRCS.2009.5251378
14. Rosen, R., von Wichert, G., Lo, G., Bettenhausen, K.D.: About the importance of autonomy and digital twins for the future of manufacturing (2015). https://doi.org/10.1016/j.ifacol.2015.06.141
15. Schmidt, N., Lüder, A.: AutomationML in a Nutshell. AutomationML - The Glue for Seamless Automation Engineering, November 2015
16. Wolf, J., Wieczorek, F., Schiller, F., Hansch, G., Wiedermann, N., Hutle, M.: Adaptive modelling for security analysis of networked control systems. In: Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016. BCS Learning & Development Ltd., Swindon, UK (2016)
17. Zhang, S., Ou, X., Homer, J.: Effective network vulnerability assessment through model abstraction. In: Holz, T., Bos, H. (eds.) DIMVA 2011. LNCS, vol. 6739, pp. 17–34. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22424-9_2