Towards formal verification of cyber security standards

Tomas Kulik <tomaskulik@eng.au.dk>
Peter Gorm Larsen <pgl@eng.au.dk>
Aarhus University, Department of Engineering
Finlandsgade 22, Aarhus, 8200, Denmark

Abstract. Cyber security standards are often used to ensure the security of industrial control systems. Nowadays, these systems are becoming more decentralized, making them more vulnerable to cyber attacks. One of the challenges of implementing cyber security standards for industrial control systems is the inability to verify early that they are compliant with the relevant standards. Cyber security standard compliance is also only validated and not formally verified, often not providing strong proofs of correct use of cyber security standard. In this paper, we propose an approach that uses formal analysis to achieve this. We formally define building blocks necessary to define the system formally in order to enable formal modeling of the system and carry out the analysis using the Alloy Analyzer. Our approach can be used at an early design stage, where problems are less expensive to correct, to ensure that the system has the desired security properties. We show the applicability of our approach by modeling two distinct cyber attacks and mitigations strategies used to defend against these attacks and also evaluate our approach based on its flexibility to handle and combine different aspects of the cyber security standards. We discuss the future directions of our research.

Keywords: cyber security; formal analysis; cyber security standards.

DOI: 10.15514/ISPRAS-2018-30(4)-5

For citation: Kulik T., Larsen P.G. Towards Formal Verification of Cyber Security Standards. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 4, 2018, pp. 79-94. DOI: 10.15514/ISPRAS-2018-30(4)-5

1. Introduction

In an industrial setting there is an increasing use of wireless technology because many components becomes Internet of Things (IoT) enabled. Rather than having to investing in a continuation of wired connections the balance between cost and agility many companies moves to such IoT solutions. However, this move towards wireless technologies gives new security challenges that must be taken serious in order to protect both the data and algorithms owned by the companies. In order to ensure this

different security standards have emerged and here the IAS/IEC-62443 is a promising candidate that deserves special examination [1].

In order to master the increase of complexity caused by the increased wireless connections the architectures of the distributed systems needs thorough analysis. Here model checking is a promising candidate to provide such an analysis. This has an appropriate balance between the time and cost spent on the analysis and the exhaustiveness favorable. In this paper we demonstrate how this can be achieved defining possible attacks and the corresponding mitigations using a formal approach. The main result is an illustration of how this kind of framework can be deployed to illustrate how a specific architecture and its chosen mitigations can be proven that the different cyber-attacks cannot be realized.

The remainder of this paper is structured as follows: Section 2 introduces the essential parts of the ISA/IEC-62443 standard and this is followed in Section 3 defining the architecture of considered system. The main result of this paper is presented in Section 4 defining extended formal framework for cyber-attacks and possible mitigations for these. Section 5 explains about how formal analysis can be conducted using the Alloy Analyzer [2]. This is followed by Section 6, which considers related work for formal analysis of cyber security standards. Finally, Section 7 provide concluding remarks also indicating the future directions planned for this work.

2. The chosen cyber security standard

Within this paper we consider security of an industrial control systems based on IoT environment. This is further considered in terms of applying cyber security standards that are used to ensure industrial automation and control system security, specifically the ISA/IEC-62443 series of standards.

The series is split into 4 distinct groups where each group considers different perspective of cyber security of the industrial automation control system (IACS). Each of the groups contain documents, where each document is understood as a single standard. This leads to name designation of specific standards based on the format: ISA/IEC-62443-*X*-*Y* where *X* is the designation of the group and *Y* is the designation of the specific document.

The first group, **ISA/IEC-62443-1, General**, considers the general aspects of the standard and cyber security. Concepts and metrics defined within this group are present throughout the other groups of the standard as shown in Fig. 1. The second group, **ISA/IEC-62443-2, Policies and procedures**, focuses on organizational aspects of cyber security. The main consideration of this group is providing the requirements that the organization has to fulfill in order to manage their cyber security program. The third group, **ISA/IEC-62443-3, System**, addresses the security on a system level. The security requirements for the system is defined here as well as guidance on implementation of these and fulfillment of these requirements. The final group, **ISA/IEC-62443-4, Component**, contains documents defining detailed requirements for cyber security on the component level.

2.1 The standard under consideration

The standard that we consider for formal verification is ISA/IEC-62443-3-3, System security requirements and security levels. This standard has been selected as it provides requirements that are applicable on system level and are verifiable by technical means. The intended audience for this standard are asset owners, system integrators and service suppliers and the purpose of this standard is to use the defined requirements to evaluate the system under consideration and determine if this system is capable of reaching a specific security level (SL). The standard defines 4 SLs:

- **SL 1**: The lowest SL aimed to prevent unauthorized disclosure of information via eavesdropping or casual exposure.
- **SL 2**: Aimed to prevent unauthorized disclosure of information to an entity actively searching for it using simple means with low resources, generic skills and low motivation.
- **SL 3**: Aimed to prevent unauthorized disclosure of information to an entity actively searching for it using sophisticated means with moderate resources, IACS specific skills and moderate motivation.
- **SL 4**: The highest SL aimed to prevent unauthorized disclosure of information to an entity actively searching for it using sophisticated means with extended resources, IACS specific skills and high motivation.

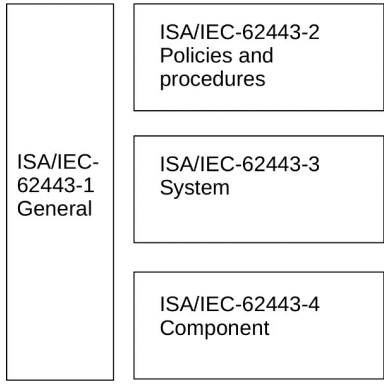

Fig. 1. Overview of ISA/IEC-62443 series structure

Within the standard the security requirements on the system level are considered as system requirements (SRs) where each SR can define 0 to 3 requirement enhancements (REs). SL of the aspect of the system is measured as a compliance with SRs and REs for this aspect, shown in Table 1.

Table. 1. Mapping between compliance with SRs, Res and corresponding SLs

SR	RE(s)	SL
SR 1	None	SL 1
SR 1	RE 1	SL 2
SR 1	RE 1 + RE 2	SL 3
SR 1	RE 1 + RE 2 + RE 3	SL 4

In case that no SR is defined for the given aspect of the system, the standard implicitly defines SL 0 as an SL for this aspect of the system.

3. System architecture

The system under consideration extends a generic control systems architecture and capabilities defined in the framework for Threat-driven Cyber Security Verification of IoT Systems (FCSVIoT) [3]. This architecture consists of subsystems equipped with sensors and actuators shown on Fig. 2. Each subsystem is a microcontroller capable of computation and communication. Communication between the subsystems creates a distributed control system, which provides data to and accepts commands from a central engineering terminal. In this paper we extend the architecture with the notion of **router**, a special type of subsystem that enables data exchange among other subsystems and extends the capabilities of the system by defining user actions on the engineering terminal. We further consider that communication channels must exist between subsystems in order to exchange data.

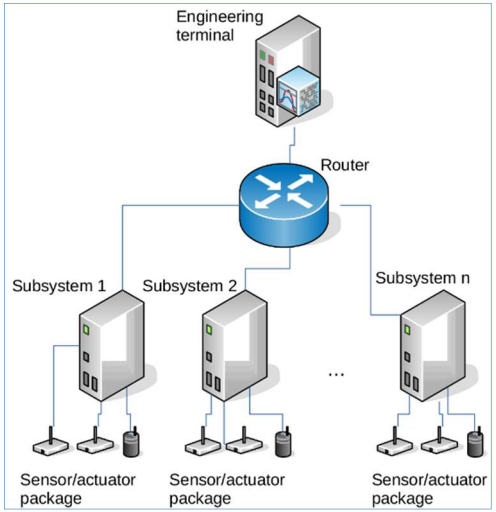

Fig. 2. Architecture of the system under consideration

We let our subsystems be governed by a set of atomic actions forming a basic alphabet for each subsystem S_i as $SA = \{generate, send, acquire, accept, discard, connect, disconnect, recover, compromise\}$ and each subsystem has a finite set of states S. Actions cause transitions between states of the subsystem such as:

$$s \xrightarrow{action(param)} s' where\ s, s' \in S$$

We further define a predicate on communication channels *secure(c)* stating that the communication channel is secured. The generate action represents generation of data by the subsystem, send action represents sending the data on a communication channel, acquire represents acquiring data from the communication channel, accept defines accepting the acquired data, discard defines discarding the acquired data. The connect and disconnect action represent a subsystem connecting to and disconnecting from a communication channel. The compromise action moves the subsystem to a compromised mode of operation, *compromised(S_i)*, where we consider that the subsystem has malicious intent. Recover action moves the subsystem from compromised to normal mode of operation, *normal(S_i)*.

We extend the actions in the FCSVIoT by considering the engineering terminal E as an user interaction part of the system by defining its own alphabet of actions $EA = \{allow, forbid\}$, where *allow* represents allowing and *forbid* represents disallowing interaction with an user by the engineering terminal. We also consider that the system holds a set of user accounts allowing users access to the Engineering terminal, *Ac* where a single account is denoted as *a*. Each account has exactly one credential *cr*, hence the system also holds a set of valid credentials *Cr*. We further define the router R as with alphabet $RA = SA \backslash \{generate\}$ as the router is not equipped with sensors to generate its own data. This leads to creation of system alphabet $A = SA \cup EA$.

4. Attacks and mitigations

We define cyber attacks as sequences of events leading to potential harm to the system under attack. Within this paper we consider two cyber attacks, specifically data packet tampering and brute force attack against an user account [4]. These attacks have been purposefully selected as the selected cyber security standard addresses them and specifies requirements for mitigations aimed to increase cost of these attacks. We provide a formal description of the attack sequence and mitigation for both of the attacks under consideration.

4.1 Data packet tampering attack

Packet tampering is the act of a compromised subsystem, specifically a router changing values in a data packet, causing the intended receiving subsystem to receive different values from those sent by the transmitting subsystem. This has then the potential to cause unsafe behavior of the system. In order to describe an instance of this attack, consider two subsystems S_0 and S_1 operating in normal mode, which we show formally as *normal(S_0)* ∧ *normal(S_1)* and a router R used to enable data

exchange between the two subsystems. The router operates in a compromised mode *compromised(R)*, meaning that some malicious actor has access to and control over this router.

The subsystems S_0 and S_1 are always connected to the router, meaning that at any time they can exchange data with the router using communication channels c_0 and c_1. We use the FCSVIoT predicate *always_connected* as *always_connected(S_0, R_0, c_0)* ∧ *always_connected(S_1, R_0, c_1)* specifying that there is always possibility of communication between S_0 and S_1 via R_0.

Now we consider that S_0 is sending a unit of data *d* to S_1. The data *d* is first obtained by R_0, which modifies the data *d*, represented by a new *modify* action added to the alphabet of the router in order to represent software installed by malicious actor, and then sends it further to S_1. The attack hence combines actions into a pattern by following a specific sequence:

1. S_0.*generate(d)*
2. S_0.*send(c_0, d)*
3. R_0.*acquire(c_0, d)*
4. R_0.*modify(d)*
5. R_0.*send(c_1, d)*
6. S_1.*acquire(c_1, d)*
7. S_1.*accept(d)*

Main act of the attack happens at the R_0.*modify(d)* event. Here the data *d* becomes malicious as *malicious(d)*. In case of non-existent mitigations within the system, the subsystem S_1 simply accepts the data and becomes itself compromised, hence the attack is successful.

In order to mitigate this attack, we consider security requirements from the ISA/IEC-62443-3-3 security standard, covering the communication integrity, namely SR 3.1 stating that *The control system shall provide the capability to protect the integrity of transmitted information*.

The requirement itself does not provide the necessary guidance on what method to use to protect the data, hence we consider the SR 3.1 RE 1 specifying the cryptographic integrity protection as *The control system shall provide the capability to employ cryptographic mechanisms to recognize changes to information during communication*. To mitigate the attack from a general perspective we consider that the data has to contain a cryptographic signature derived from the data content and a secret known to subsystems, but not routers. This introduces a concept of signed data, which we do by extending the alphabet of the subsystem by adding an atomic *sign* event as *sign(d)*. We further define a predicate for signed data stating that the data is considered signed only if signed by a subsystem operating in a normal mode:

$$signed(d) = \exists s: s \xrightarrow{sign(d)}_i s' \wedge s \in S_N$$

We then consider applying the signing event as a mitigation by specifying that the subsystem discards the signed data if it has been modified, with indices added to the state notation, describing the order of state transitions:

$$\forall\, s_1 : s_1 \xrightarrow{discard(d)}_i s_2 \text{ if}$$
$$always\_connected(S_i, R_j, c_k)$$
$$\text{and } \exists s_0 : s_0 \xrightarrow{acquire(d,c_k)}_i s_1$$
$$\text{and } \exists s : s \xrightarrow{modify(d)}_j s'$$
$$\text{and } signed(d)$$

Applying this mitigation by adding S_0.*sign(d)* after the S_0.*generate(d)* event causes the final event in the chain to be S_1.*discard(d)* since R_0.*modify(d)* is present. This means that the subsystem S_1 does not enter compromised mode and the cyber attack is unsuccessful.

4.2 Brute force attack against an user account

Brute force attack against an user account uses computational power to try to guess user sign in credentials by randomly generating passwords and user names and providing them to the system for verification. The attack can be streamlined if the user name and length of the password is known, decreasing the "guess space", which in turn leads to less time required to guess the correct credentials. If the user account can be breached this gives the malicious actor control over the system in terms that the breached account allows, potentially allowing the malicious actor submission of malicious commands to the system. To formally describe the attack we consider a single engineering terminal E_0 operating in a normal mode, *normal*(E_0). We further define a *check* function on an engineering terminal, responsible for raising the *allow* or *forbid* event:

$$check(cr) = \begin{cases} allow, & \text{if } cr \in Cr \\ forbid, & \text{otherwise} \end{cases}$$

We formulate the attack as a recursive *crack(cr)* function that generates new *cr* for every attempt used to find a *cr* such that *cr* ∈ *Cr*:

$$crack(check(cr)) = \begin{cases} true, & \text{if } allow \\ crack\left(check\left(new(cr)\right)\right), & \text{if } forbid \end{cases}$$

Once the function returns true the malicious actor has obtained access to the engineering terminal, causing the engineering terminal to operate in a compromised mode of operation, as *compromised*(E_0) and the attack is considered successful.

In order to mitigate the attack we consider the requirement SR 1.11 defined in ISA/IEC-62443-3-3, stating, *The control system shall provide the capability to enforce a limit of a configurable number of consecutive invalid access attempts by any user (human, software process or device) during a configurable time period. The control system shall provide the capability to deny access for a specified period of time or until unlocked by an administrator when this limit has been exceeded*. To enforce this we define a locked predicate acting on specific account mapped via its

valid credential where mapping between account *ac* and credential *cr* is one to one and hence for simplicity we omit *cr* and consider *ac* as belonging to a specific *cr* as:

$$locked(ac) = \neg\, \exists s : s \xrightarrow{allow()} s' : ac \in Ac$$

We then need to consider the amount of allowed invalid access attempts. In order to abstract away from details of password complexity, we present an assumption stating that the successful brute force attack against an system that allows reasonable small amount (in general we would consider this less than 10 for practical reasons) invalid access attempts is so unlikely that we consider it impossible. Using this assumption as a mitigation we can guarantee that the user account cannot be breached by brute force attack. We also abstract away from notion of time intervals as we consider that the brute force attack is happening rapidly and would always exceed the amount of tries within a specific time interval. We formally show this mitigation by first defining a global variable for *ac* holding the current attempt as *attempt(ac)* for its credential *cr*:

$$attempt(ac) = \begin{cases} attempt(ac) + 1, & \text{if } check(cr) = forbid \\ 0, & \text{otherwise} \end{cases}$$

We then use the variable in adding an attempt limit on using a credential to sign in to an user account such that the account becomes locked if the maximum amount of attempts is reached:

$$limited(cr, max\_att) = locked(ac) \text{ if } attempt \geq max\_att$$

By applying the *limited* predicate to the credentials we cause the account to become *locked* as a result of the *crack* function. Since a locked account cannot be used to gain access to the engineering terminal, the cyber attack fails and the engineering terminal continues in the normal mode of operation, *normal*(E_0). It is important to note that in general the *max_att* has to be set in such a way that does not hinder usability of the system, while providing assurance of sufficient security. This mitigation strategy has therefore a limitation in case the *max_att* is set unreasonably large.

5. Formal analysis

In this section we shortly present the extensions made to the FCSVIoT and show how the mitigations for data packet tampering and brute force against user account attacks have been verified when considered within the architecture defined in Section 2. This is achieved by expressing the aforementioned attacks and mitigations using FCSVIoT with extensions introduced in this paper and verifying these scenarios using the Alloy Analyzer.

5.1 Short introduction to Alloy Analyzer

Alloy is a formal specification language, based on first order logic, used for expressing structural constraints in software systems. Alloy allows for modeling at different levels of abstraction, where at the highest level it provides object oriented interpretation, at second level it uses the set theory and at the lowest level atoms and relations are used. Within our model we are using the set theory, atoms and relations

to model the types using the **sig** keyword. Subtyping is supported in Alloy by usage of **extends** keyword. We model relations between objects by specifying mappings between sets, for example `has:set EngTerminal one->some Account`, where `has` is the relation stating that the **one**, meaning exactly one engineering terminal has **some**, meaning at least one account associated with it. The scope of the model is specified after the **run** block, by quantifying how many atoms do we want to include in the model by using the **exactly** keyword. Properties of the Alloy model can be verified by usage of the Alloy Analyzer software tool [5], which checks properties of the model by generating counterexamples.

5.2 Overview of extensions to FCSVIoT

Among the first extensions is addition of new data types `Router` corresponding to the router, `EngTerminal` corresponding to the engineering terminal, `Account` corresponding to user account and `Credential` corresponding to credential for specific account as specified in Section 3. Using Alloy Analyzer, we define these datatypes using set definitions, represented by the **sig** keyword. We further extend the `State` definition with number of new relations. We further define the router as an extension of `Device` type and also adapt `Subsystem` to be extension of `Device`, as shown in Listing 1.

The `Device` type is used as a base type since `Router` and `Subsystem` share most of the actions. The only difference is that we consider that the router is not capable of generating data. The relations within `State` now also use `Device` in order to model relation that cover both `Router` and `Subsystem`. For example, the `compromised` relation shown in Listing 1 shows that a state can contain any number of compromised devices. Another relation recorded in each state is for example `accepted` which maps devices to data.

The different types discussed above are governed by several facts, which are understood as constraints on the model. One of these is the consideration that the data is either signed or not and this does not change as the system progresses in its state transitions. This is shown in Listing 2. In this constraint `s'` is the state following the `s` state, hence the constraint guarantees that the data remains signed in all states.


```
open util/ordering[State]
sig Data {}
sig Device {}
sig Subsystem extends Device{}
sig Router extends Device{}
sig Channel {}
sig EngTerminal {}
sig Credential{}
sig Account{}
sig State {
...
compromised: set Device,
can_authorise: set Subsystem,
malicious: set Data,
signed: set Data,
accepted: set Device -> set Data,
secure: set Channel,
attempts_exceeded: set Account,
limited: set Account,
cracked: set Account,
large: set Credential,
locked: set Account,
has:set EngTerminal one->some Account,
hasCred:set Account one->one Credential
...
}{ /* Facts belonging to State */ ... }
```

Listing 1. Extensions and changes to the modeling framework

```
fact{
...
all s:State, s':s.next |
s.signed = s'.signed
}
```

Listing 2. Global constraint governing signed data

5.3 Verification of data tampering mitigation strategy

Here we demonstrate the mitigation strategy applied to a scenario discussed in 4.1. The simplest model to demonstrate data tampering mitigation strategy in fact only requires one subsystem and a router as it is the router that is responsible for the attack. This is shown in Listing 3. The listing shows the constraint for mitigation and the setup of the model. The complete extended FCSVIoT can be found via [6].

```
run {
...
// mitigation signed data
all s:State | all d:Data | d in s.signed
//test the condition
some malicious
...
} for
exactly 5 State
, exactly 1 Subsystem
, exactly 1 Data
, exactly 1 Channel
, exactly 1 Router
, exactly 2 Device
, exactly 0 EngTerminal
...
```

Listing 3. Verifying the data tampering mitigation strategy using Alloy

The **run** commands checks that the data is signed in five states, required to exectute the whole scenario. The result of this execution is: `No instance found`. This means that the Alloy Analyzer could not find a counter-example within the requested scope and the mitigation strategy is proven to work.

5.4 Verification of brute force attack:

We will show how the mitigation strategy for brute force attack can be modeled by considering a scenario described in Section 4.2. The **run** command for the model we use consists of one engineering terminal with one user account, with its associated credentials and omits subsystems as shown in Listing 4. The mitigation used in this scenario states that all accounts within all states of the system are always considered limited (i.e. they consider limit on the number of unsuccessful login attempts).

This scenario considers three states, creating the smallest scope necessary for its execution. Once the **run** command is executed the Alloy Analyzer returns `No instance found`, confirming that the mitigation strategy prevents the user account from being cracked.


```
run {
...
// mitigation account has limited tries
all s:State | all a:Account |
a in s.limited
//test the condition
some cracked
//start with no cracked account
no first.cracked
...
}for
exactly 3 State
, exactly 1 EngTerminal
, exactly 1 Account
, exactly 1 Credential
, exactly 0 Subsystem
...
```

Listing 4. Verifying the brute force attack mitigation strategy using Alloy

6. Related work

As cyber security is becoming very important topic in the industry, mainly in advent of digitalization and trends such as industry 4.0 [7], research is being carried out within the area of using formal methods in order to provide proofs that systems meet cyber security requirements [8][9][10]. The benefits of using model based verification are its applicability at an early stage of system development in order to help avoid exposure to attacks as well as provide mitigations for attacks that are not easily avoidable [11][12]. This approach consists of formal description of the behavior of a system and formal description of cyber-attacks and mitigations. The complete model is then formally analyzed in order to verify that the mitigation strategies prevent the cyber-attacks from causing potentially harmful behavior of the system. Sometimes specific cyber security standards are considered as criteria for these mitigations strategies [13].

In order to provide assurance that an industrial control system meets criteria specified in a cyber security standard, authors of [14] have investigated the ISA-99.01-01 standard by considering the requirements and metrics specified within the standard. While the authors have described part of the standard formally, their goal was not to conduct formal analysis to ensure the satisfiability of the security requirements by a given architecture but rather to provide recommendations to the operators of industrial control systems to not blindly trust standards but verify their security impact on the system.

The authors of [15] have proposed a formalization and verification technique for ISO/IEC-15408 standard known as Common Criteria using Z notation. In their

technique they consider the natural language definitions within the standard and create formal templates based on these. The authors suggest usage of the templates against the formalized specification of the target system, which is left to the party verifying the system against the instantiated templates. The authors provide an example of this verification using the Z/EVES theorem prover. Our approach differs by providing formal building blocks for the system from the start, hence formalization of the system can be done by selecting from these building blocks.

7. Conclusions and future work

So far this research has demonstrated that the chosen approach is quite extensible where this paper has demonstrated how the models made in Alloy can be extended in a conservative manner with additional threats. It is expected that we in the future in this context is furthering the formal definitions to encompass more of aspects of the security standard and to verify these against larger variety of cyber-attacks. We further consider switching to TLA+ [16] in order to show the applicability of our framework using different formalism.

Acknowledgments

This work is partially supported by the Manufacturing Academy of Denmark (MADE) Digital project. For more information see http://www.made.dk/.

References

[1]. International Society of Automation. The 62443 Series of Standards. http://isa99.isa.org/Public/Information/The-62443-Series-Overview.pdf, accessed on 13/3/18

[2]. D. Jackson, Software Abstractions: Logic, Language, and Analysis. Heyward Street, Cambridge, MIT Press, April 2006, iSBN-10: 0-262-10114-9.

[3]. Tomas Kulik, Peter W. V. Tran-Jørgensen, Jalil Boudjadar, and Carl Schultz. A framework for threat-driven cyber security verification of iot systems. In Proc. of the First International Workshop on Verification and Validation of Internet of Things, Västerås, Sweden, april 2018, in print.

[4]. C. Bekara. Security issues and challenges for the iot-based smart grid. Procedia Computer Science, vol. 34, 2014, pp. 532–537. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914009193

[5]. The Alloy Analyzer Modelling website, http://alloy.mit.edu/alloy/, 2018

[6]. Tomas Kulik and Peter Gorm Larsen. Extensions to formal security modeling framework. https://github.com/kuliktomas/FCSVIoT/commit/189c7962f7f0870fa5315c31a71a6b35e896e47d, 2018.

[7]. N. Jazdi. Cyber physical systems in the context of industry 4.0. In Proc. of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, May 2014, pp. 1–4.

[8]. M. Ge and D. S. Kim. A framework for modeling and assessing security of the internet of things. In Proc. of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), 2015, pp. 776–781.

[9]. C. Heitmeyer, M. Archer, E. Leonard, and J. McLean. Applying Formal Methods to a Certifiably Secure Software System. Software Engineering, IEEE Transactions on Software Engineering, vol. 34, no. 1, 2008, pp. 82 –98.

[10]. A. N. Haidar and A. E. Abdallah. Formal modelling of pki based authentication. Electronic Notes in Theoretical Computer Science, vol. 235, 2009, pp. 55 – 70. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S157106610900084X

[11]. D. C. Wardell, R. F. Mills, G. L. Peterson, and M. E. Oxley. A method for revealing and addressing security vulnerabilities in cyber-physical systems by modeling malicious agent interactions with formal verification. Procedia Computer Science, vol. 95, no. Supplement C, 2016, pp. 24 – 31, Available: http://www.sciencedirect.com/science/article/pii/S1877050916324619

[12]. F. A. Teixeira, F. M. Pereira, H.-C. Wong, J. M. Nogueira, and L. B. Oliveira. Siot: Securing internet of things through distributed systems analysis. Future Generation Computer Systems, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17304235

[13]. J. Woodcock, S. Stepney, D. Cooper, J. A. Clark, and J. Jacob. The Certification of the Mondex Electronic Purse to ITSEC Level E6. Formal Aspects of Computing, vol. 20, no. 1, 2008, pp. 5–19.

[14]. D. K. Holstein and K. Stouffer. Trust but verify critical infrastructure cyber security solutions. In Proc. of the 43rd Hawaii International Conference on System Sciences, 2010, pp. 1–8.

[15]. S. Morimoto, S. Shigematsu, Y. Goto, and J. Cheng. Formal verification of security specifications with common criteria. In Proceedings of the 2007 ACM Symposium on Applied Computing, ser. SAC '07, 2007, pp. 1506–1512. [Online]. Available: http://doi.acm.org/10.1145/1244002.1244325

[16]. L. Lamport, Specifying systems: the TLA+ language and tools for hardware and software engineers. Addison-Wesley Longman Publishing Co., Inc., 2002, Boston, MA, USA, 384 pages.

К формальной верификации стандартов кибербезопасности

Томаш Кулик <tomaskulik@eng.au.dk>
Питер Горм Ларсен <pgl@eng.au.dk>
Орхусский университет, Департамент инженерии,
Finlandsgade 22, Орхус, 8200, Дания

Аннотация. Стандарты кибербезопасности часто используются для обеспечения защищенности промышленных систем управления. В последнее время такие системы становятся все более децентрализованными, что делает их все более уязвимыми для разного рода кибератак. Одна из проблем реализации стандартов кибербезопасности в промышленных системах управления состоит в том, что невозможно своевременно проверить, соответствуют ли разрабатываемые системы этим стандартам или нет. Помимо прочего, соответствие стандарту кибербезопасности только валидируется, а не верифицируется формально, что, как правило, не дает убедительных доказательств правильного использования стандарта. В статье предлагается подход, в котором проверка защищенности промышленных систем управления осуществляется путем формального анализа. Подход состоит в следующем: определяются строительные блоки, необходимые для формального описания системы; составляется формальная модель системы; модель анализируется с помощью инструмента Alloy Analyzer. Предлагаемый подход может использоваться на ранних стадиях проектирования, где проблемы не так дороги для исправления. Чтобы показать применимость подхода, были смоделированы две кибератаки, а также стратегии противодействия им. Подход был также оценен на предмет гибкости — возможности совмещения разных аспектов стандартов кибербезопасности. В статье также обсуждаются будущие направления исследования.

Ключевые слова: кибербезопасность, формальный анализ; стандарты кибербезопасности

DOI: 10.15514/ISPRAS-2018-30(4)-5

Для цитирования: Кулик Т., Ларсен П.Г. К формальной верификации стандартов кибербезопасности. Труды ИСП РАН, том 30, вып. 4, 2018 г., стр. 79-94 (на английском языке). DOI: 10.15514/ISPRAS-2018-30(4)-5

Список литературы

[1]. International Society of Automation. The 62443 Series of Standards. http://isa99.isa.org/Public/Information/The-62443-Series-Overview.pdf, accessed on 13/3/18

[2]. D. Jackson, Software Abstractions: Logic, Language, and Analysis. Heyward Street, Cambridge, MIT Press, April 2006, iSBN-10: 0-262-10114-9.

[3]. Tomas Kulik, Peter W. V. Tran-Jørgensen, Jalil Boudjadar, and Carl Schultz. A framework for threat-driven cyber security verification of iot systems. In Proc. of the First International Workshop on Verification and Validation of Internet of Things, Västerås, Sweden, april 2018, in print.

[4]. C. Bekara. Security issues and challenges for the iot-based smart grid. Procedia Computer Science, vol. 34, 2014, pp. 532–537. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914009193

[5]. The Alloy Analyzer Modelling website, http://alloy.mit.edu/alloy/, 2018

[6]. Tomas Kulik and Peter Gorm Larsen. Extensions to formal security modeling framework. https://github.com/kuliktomas/FCSVIoT/commit/189c7962f7f0870fa5315c31a71a6b35e896e47d, 2018.

[7]. N. Jazdi. Cyber physical systems in the context of industry 4.0. In Proc. of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, May 2014, pp. 1–4.

[8]. M. Ge and D. S. Kim. A framework for modeling and assessing security of the internet of things. In Proc. of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), 2015, pp. 776–781.

[9]. C. Heitmeyer, M. Archer, E. Leonard, and J. McLean. Applying Formal Methods to a Certifiably Secure Software System. Software Engineering, IEEE Transactions on Software Engineering, vol. 34, no. 1, 2008, pp. 82 –98.

[10]. A. N. Haidar and A. E. Abdallah. Formal modelling of pki based authentication. Electronic Notes in Theoretical Computer Science, vol. 235, 2009, pp. 55 – 70. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S157106610900084X

[11]. D. C. Wardell, R. F. Mills, G. L. Peterson, and M. E. Oxley. A method for revealing and addressing security vulnerabilities in cyber-physical systems by modeling malicious agent interactions with formal verification. Procedia Computer Science, vol. 95, no. Supplement C, 2016, pp. 24 – 31, Available: http://www.sciencedirect.com/science/article/pii/S1877050916324619

[12]. F. A. Teixeira, F. M. Pereira, H.-C. Wong, J. M. Nogueira, and L. B. Oliveira. Siot: Securing internet of things through distributed systems analysis. Future Generation Computer Systems, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17304235

[13]. J. Woodcock, S. Stepney, D. Cooper, J. A. Clark, and J. Jacob. The Certification of the Mondex Electronic Purse to ITSEC Level E6. Formal Aspects of Computing, vol. 20, no. 1, 2008, pp. 5–19.

[14]. D. K. Holstein and K. Stouffer. Trust but verify critical infrastructure cyber security solutions. In Proc. of the 43rd Hawaii International Conference on System Sciences, 2010, pp. 1–8.

[15]. S. Morimoto, S. Shigematsu, Y. Goto, and J. Cheng. Formal verification of security specifications with common criteria. In Proceedings of the 2007 ACM Symposium on Applied Computing, ser. SAC '07, 2007, pp. 1506–1512. [Online]. Available: http://doi.acm.org/10.1145/1244002.1244325

[16]. L. Lamport, Specifying systems: the TLA+ language and tools for hardware and software engineers. Addison-Wesley Longman Publishing Co., Inc., 2002, Boston, MA, USA, 384 pages.