Security Evaluation of Cyber-Physical Systems Using Automatically Generated Attack Trees

Laurens Lemaire[1(✉)], Jan Vossaert[1], Bart De Decker[2], and Vincent Naessens[1]

[1] MSEC, iMinds-DistriNet, Department of Computer Science, KU Leuven, Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
{laurens.lemaire,jan.vossaert,vincent.naessens}@cs.kuleuven.be
[2] iMinds-DistriNet, Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
bart.decker@cs.kuleuven.be

Abstract. The security of cyber-physical systems (CPS) is often lacking. This abstract presents a methodology that performs a security evaluation of these systems by automatically generating attack trees based on the system model. The assessor can define different kinds of attackers and see how the attack tree is evaluated with respect to a specific type of attacker. Optimal attacker strategies are calculated and from here the most vulnerable elements of the system can be derived.

Keywords: Cyber-physical systems · Attack trees
Security assessment

1 Introduction

Cyber-physical systems (CPS) are networks of interacting elements with physical input and output, usually containing various remote field sites where a certain process is taking place [4]. Each field site consists of sensors and actuators, controlled locally by a programmable logic controller (PLC) or similar device. These remote sites are connected to a centralized control network where operators can remotely monitor and control the processes. In the past decades, these systems have evolved from proprietary, isolated systems to complex interconnected systems that are remotely accessible and often use commercial off-the-shelf (COTS) components. This has made them easier to use, but also easier to attack [2].

Various research initiatives have been undertaken in previous years to improve the security of cyber-physical systems, both in the academic world and by industry. One of the areas in which a lot of research is situated is risk assessment. A risk assessment process evaluates the likelihood and impact of identified threats to the CPS. A recent review of risk assessment methodologies for CPS lists various remaining research challenges [1], among which are better tool support, the necessity of proper methodology validation, more attention towards the system architecture when performing a risk assessment, and the need for reliable sources of probabilities. In this abstract, a methodology for the security evaluation of CPS is presented which can be used in a risk assessment process and which tackles the aforementioned challenges.

2 Methodology

Figure 1 shows a general overview of the methodology. The process is divided in two parts: a *tree generation algorithm* and a *tree evaluation algorithm*. The tree generation algorithm takes as input a *system*, a list of extracted system *vulnerabilities* and an *attacker goal*. The *system* is a model of the CPS which will be subject to the security evaluation. The *vulnerabilities* themselves are automatically derived from the system model by the FAST-CPS framework [5]. The *attacker goal* is the global objective of the attacker with regards to the CPS. For each attacker goal for a system, a separate attack tree will be built. *Templates* are used by the tree generation algorithm, they generate parts of the tree. The templates are part of the methodology and do not require human interaction. Once the attack tree is generated, the assessor can provide an *attacker* as input for the tree evaluation algorithm. The *attacker* is an entity that attempts to reach the attacker goal by following a path of attack steps through the attack tree. An attacker is defined in terms of his capabilities, the credentials he possesses and the parts of the system he has physical access to. Different kinds of attackers can be modelled, each with their own capabilities. The tree evaluation algorithm results in an *optimal attacker strategy* and the *difficulty* of the attack.

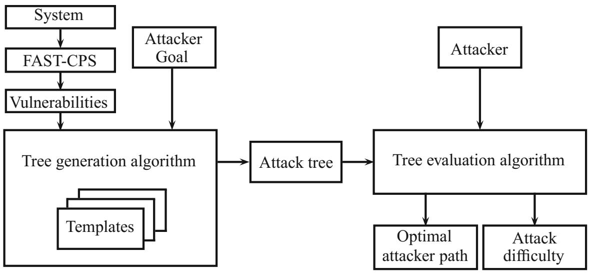

Fig. 1. The flow of the methodology.

2.1 Input Model

The assessor must provide three input elements: the system model, the attacker goal, and the type of attacker.

System Model. The attack trees are based on the system architecture, hence the assessor must provide a *system model*. An existing framework can be used for this task: FAST-CPS. This framework allows the assessor to model his system architecture in SysML, a modelling language derived from UML used for model-based systems engineering [3].

Attacker Goal. The *attacker goal* will become the root node of the attack tree. The assessor chooses this goal, the rest of the tree is then automatically generated. If an assessor wants to reason about multiple attacker goals, a separate tree is built for each goal. Three main types of attacker goals were identified based on the NIST guide to ICS security [6]:

– *Modify(Parameter)*: Changing the process behaviour of the CPS. The assessor must provide which process parameter the attacker wants to modify.
– *DenialOfService(SystemPart)*: Halting the workings of a component or module in the CPS.
– *Obtain(Asset)*: Obtaining a data asset stored in the CPS.

Attacker. An Attacker $\mathcal{A} = (C, M, A)$ is defined by the set of credentials "C" the attacker owns, a mapping of *attacker capabilities* to integer values "M", and a set of components in the system "A" which the attacker has physical access to.

2.2 Output

The output of the methodology is an optimal attacker strategy. An *Optimal Attacker Strategy S* is a sub-tree of the attack tree T. This sub-tree represents the optimal way for a modelled attacker to reach the attacker goal. What is optimal depends on the chosen heuristic in the tree evaluation stage. Once the optimal attacker strategy is returned, the assessor can identify the most vulnerable parts of the system by looking at the leaves of S. Once the system model has been changed accordingly, the assessor can see if the optimal attacker strategy has changed and which other system parts require attention.

2.3 Tree Generation

The tree generation algorithm takes the attacker goal and a CPS model and generates an attack tree. The algorithm uses templates that represent attacks on the system or assess the impact of vulnerable components on possible threats. The algorithm is shown below:

```
GenerateTree(AttackerGoal,SystemModel,Templates):

tree.root = AttackerGoal
goals = { AttackerGoal }
while goals $\neq \emptyset$ do
    goal = goals.pop()
```

```
foreach Template t ∈ Templates do
    if t.goal = goal then
        Tree s = t.execute(goal, SystemModel)
        goals.push(s.leaves)
        tree.replace(goal, s)
```

Initially there is only one goal in the set of goals, which is the attacker goal chosen by the assessor. Each template has a unique goal associated with it, once the template that matches the attacker goal is found, this template is executed and a tree is generated. This tree is then added to the main tree where the goal used to be. Each template has a specific execute method. The leaves of an executed template are then added to the set of goals so they can be matched with other templates. If a goal finds no match, it is a leaf of the final tree.

2.4 Tree Evaluation

Once a possible attack on a system has been modelled in an attack tree, the tree can be used to analyse security properties of the system, for instance the difficulty of attacks. The analysis proceeds in three steps: First the difficulty of each leaf is determined, then the difficulty of parent nodes is synthesized from the difficulty of their children. Once all nodes are annotated, the optimal attacker strategy is calculated.

3 Conclusions

This abstract presents a methodology that automatically builds system-dependent attack trees for the security evaluation of cyber-physical systems. The assessor can model different types of attackers and evaluate the attack tree with respect to a specific type of attacker. Optimal attacker strategies are calculated during this evaluation phase.

References

1. Cherdantseva, Y., et al.: A review of cyber security risk assessment methods for scada systems. Comput. Secur. **56**, 1–27 (2016)
2. ENISA. Protecting industrial control systems: Recommendations for Europe and member states (2011)
3. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann, Burlington (2014)
4. Lee, E.A.: Cyber physical systems: design challenges. In: 2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), pp. 363–369. IEEE (2008)
5. Lemaire, L., Lapon, J., De Decker, B., Naessens, V.: A SysML extension for security analysis of industrial control systems. In: Proceedings of the 2nd International Symposium for ICS & SCADA Cyber Security Research, p. 1 (2014)
6. Stouffer, K., Lightman, S., Pillitteri, V., Abrams, M., Hahn, A.: Guide to industrial control systems (ICS) security (2015)