CyberSAGE: A Tool for Automatic Security Assessment of Cyber-Physical Systems

An Hoa Vu[1], Nils Ole Tippenhauer[1], Binbin Chen[1]
David M. Nicol[2], and Zbigniew Kalbarczyk[2]

[1] Advanced Digital Sciences Center, Singapore
[2] University of Illinois at Urbana-Champaign, IL, USA

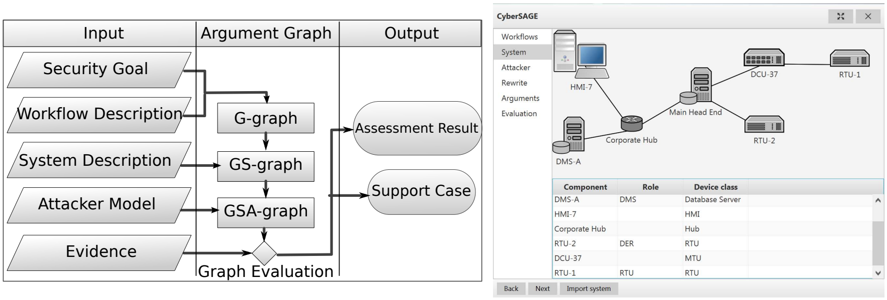
Abstract. We present *CyberSAGE*, a Cyber Security Argument Graph Evaluation tool for cyber-physical systems. Specifically, CyberSAGE supports the automatic generation of *security argument graphs*, a graphical formalism that integrates diverse inputs—including workflow information for processes executed in the system, physical network topology, and attacker models—to argue about the level of security for the target system. Based on the generated graphs, CyberSAGE can combine numerical information to compute quantitative security assessment results. We illustrate the use of CyberSAGE through a power grid case study.

1 Introduction

Assessing the security of cyber-physical systems (CPS) in a holistic manner is challenging, since the results depend on a wide range of heterogeneous inputs: how the system is used, its network topology, which types of possible attacks one should consider, etc. In our previous work [1], we proposed a CPS security assessment framework that uses workflow—describing how a system provides its intended functionality—as a pillar for organizing different inputs. As shown in Figure 1a, our proposed framework suggests to first use the information about a security goal and the related workflow description to generate a high-level goal graph called *G-graph*, which can then be be used to generate a *GS-graph* by incorporating system information and finally a *GSA-graph* by further adding attacker information. We call the generated structures security argument graphs—they provide a graphical formalism that integrates diverse pieces of security-related inputs to argue about the security of the target system (more details in [2]). The graphs also support the combination of different pieces of numerical evidence (associated with different inputs) to produce quantitative assessment results.

While it is easy to explain the intuition behind the process, the manual construction of a holistic security argument graph for a complex CPS can be costly and error-prone. To better deal with the complexity, we have developed *CyberSAGE*, a Cyber Security Argument Graph Evaluation tool for CPS security assessment. Though still in its prototype stage, CyberSAGE can already automatically generate security argument graphs by putting together different types of inputs according to our methodology. It also supports a combinatorial

G. Norman and W. Sanders (Eds.): QEST 2014, LNCS 8657, pp. 384–387, 2014.



Component	Role	Device class
DMS-A	DMS	Database Server
HMI-7		HMI
Corporate Hub		Hub
RTU-2	DER	RTU
DCU-37		MTU
RTU-1	RTU	RTU

(a)

(b)

Fig. 1. The assessment framework implemented by CyberSAGE and its snapshot

approach to compute quantitative metrics over the graph. Figure 1b shows a snapshot of CyberSAGE. The rest of this paper will describe its main functionalities and illustrates its use in an example case study. More information about CyberSAGE can be found at our tool website [3].

2 Use of CyberSAGE

CyberSAGE can automatically evaluate a security goal that relates to the availability of specific processes. Those processes model the intended physical, cyber, and human interactions in the target CPS, and are provided to CyberSAGE as XML-based specifications. CyberSAGE converts the XML-based input into internal data structures and uses them to generate a security argument graph based on predefined *extension templates*. These templates are described in more details in [2], together with their definition and a set of CPS-specific templates. CyberSAGE performs the overall evaluation process in the following stages:

1) Goal and workflow information input stage. This stage loads the workflow for which the availability will be assessed. Since the workflow is typically modeled using UML activity diagrams, CyberSAGE supports XMI format inputs, as produced by UML modeling tools like Enterprise Architect[1].

2) System information input stage. This stage collects information about the deployed system. Currently, CyberSAGE can parse the topology information about a network, where each device plays one or more roles corresponding to the actors in the workflows. Devices are associated with properties such as availability, vulnerabilities, etc, according to their classes. CyberSAGE supports system inputs in an XML dialect used by the CSET tool [4].

3) Attacker information input stage. The next stage involves modeling potential threats to the system. Our attacker model contains a list of potential attack actions for different device classes and properties, and the required attacker properties to perform those actions. Currently, CyberSAGE has modeled

[1] `http://www.sparxsystems.com/products/ea/`

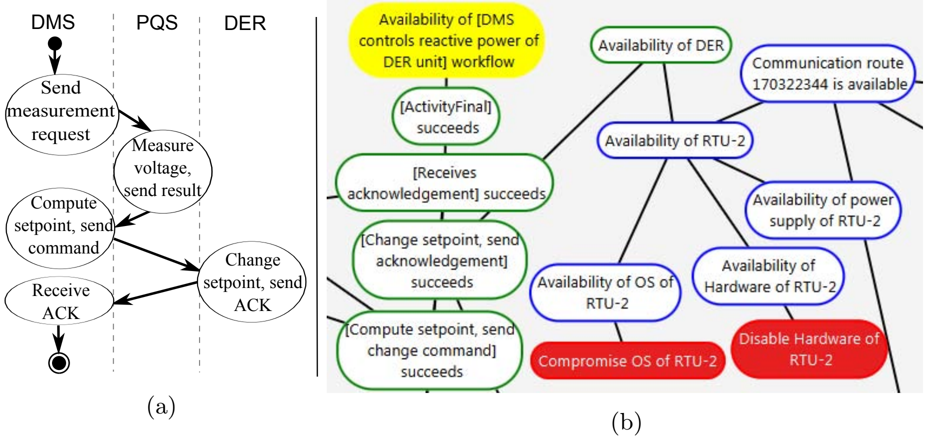

DMS
PQS
DER
Send measurement request
Measure voltage, send result
Compute setpoint, send command
Change setpoint, send ACK
Receive ACK
(a)
Availability of [DMS controls reactive power of DER unit] workflow
[ActivityFinal] succeeds
[Receives acknowledgement] succeeds
[Change setpoint, send acknowledgement] succeeds
[Compute setpoint, send change command] succeeds
Availability of DER
Communication route 170322344 is available
Availability of RTU-2
Availability of power supply of RTU-2
Availability of OS of RTU-2
Availability of Hardware of RTU-2
Compromise OS of RTU-2
Disable Hardware of RTU-2
(b)

Fig. 2. The workflow input (a) and generated security argument graph in CyberSAGE (b, partially shown) for the example use case

attack actions that are targeted at the availability of software or hardware components in the system, via either remote or local access to the devices.

4) Argument graph generation stage. CyberSAGE then generates the argument graph using a built-in set of CPS-specific extension templates as defined in [2]. Users can inspect the generated argument graph and return to previous stages to change inputs and subsets of extension templates.

5) Evaluation stage. This stage performs a quantitative evaluation of the constructed security argument graph. Currently, CyberSAGE supports the labeling of the vertices by numerical evidence including component availability (when not under attack) and attack success probability, as well as the `AND`, `OR`, `NEGATION` operations for combining evidence. It then invokes the external libDAI [5] with a transformed form of the graph to compute the availability of the concerned process through Bayesian evaluation.

Example Use Case. We have used CyberSAGE in multiple use cases to assess the availability property of various CPS under attack. Due to space limitations, we focus on a concrete distributed energy resources control example (as adapted from [6]).

In the use case, the considered workflow (Figure 2a) captures the interactions among three main actors: a distribution management system (DMS) that manages the power quality and stability of a power grid; distributed energy resources (DER), such as solar power generators, that can adapt power generation based on the request from DMS; and a power quality sensor (PQS) that measures various power quality indicators, e.g., the voltage, and reports them to DMS. On a high level, the DMS controls the power generation output of DER based on the measurements from PQS. These three actors are implemented by distributed physical components, e.g., remote terminal units (RTU), that are not directly connected to each other. The system topology input captures the connectivity between the different physical components. Finally, we consider different types of attacks on different components and assign numerical evidence for the attack probability and component availability.

CyberSAGE applies a set of predefined extension templates [2] to incorporate the above inputs and generate a security argument graph, which consists of 42 vertices, as (partially) shown in Figure 2b. To interpret the graph, its root shows the security goal, and each vertex is expanded to one or several other vertices that it depends on. Based on the graph and numerical information provided at its vertices, CyberSAGE computes the availability of the modelled process. The runtime needed for generating the graph and evaluating the result is about 40ms.

We also tested other use cases with CyberSAGE, where the largest case had a security argument graph of 163 vertices and incurred a runtime of around 200ms. Since a security argument graph is meant to be human-readable (hence likely has no more than a few hundreds of vertices), we do not expect CyberSAGE to have performance issues for its graph generation and combinatorial computation.

3 Conclusion and Acknowledgements

In this paper, we introduced CyberSAGE, a tool that implements our workflow-oriented security assessment framework [1]. CyberSAGE supports automatic generation of security argument graphs and quantitative security assessment of CPS based on the generated graphs. We demonstrate how to use CyberSAGE to conduct an automatic security assessment for an electrical power grid use case.

This work is supported by Singapore's Agency for Science, Technology, and Research (A*STAR) under the Human Sixth Sense Programme (HSSP). We thank QEST reviewers and our shepherd David Parker for useful feedback, and William Temple, Sumeet Jauhar, and William Sanders for helpful discussions.

References

1. Chen, B., Kalbarczyk, Z., Nicol, D.M., Sanders, W.H., Tan, R., Temple, W.G., Tippenhauer, N.O., Vu, A.H., Yau, D.K.: Go with the flow: Toward workflow-oriented security assessment. In: New Security Paradigms Workshop (2013)
2. Tippenhauer, N.O., Temple, W.G., Vu, A.H., Chen, B., Nicol, D.M., Kalbarczyk, Z., Sanders, W.H.: Automatic generation of security argument graphs. Technical Report 1405.7475, CoRR (2014)
3. CyberSAGE: Tool Website, `http://cybersagetool.com`
4. CSET: The cyber security evaluation tool,
   `http://ics-cert.us-cert.gov/satool.html`
5. Mooij, J.M.: libDAI: A free and open source C++ library for discrete approximate inference in graphical models. Journal of Machine Learning Research 11, 2169–2173 (2010)
6. CEN-CENELEC-ETSI Smart Grid Coordination Group: Smart grid reference architecture (November 2012), `http://ec.europa.eu/energy/gas_electricity/smartgrids/doc/xpert_group1_reference_architecture.pdf`