A Study on Mitigation Techniques for SCADA-Driven Cyber-Physical Systems (Position Paper)

Mariana Segovia[1], Ana Rosa Cavalli[1], Nora Cuppens[2], and Joaquin Garcia-Alfaro[1(✉)]

[1] Télécom SudParis, Evry, France
{segovia,ana.cavalli,joaquin.garcia_alfaro}@telecom-sudparis.eu
[2] IMT Atlantique, Cesson Sévigné, France
nora.cuppens@imt-atlantique.fr

Abstract. Cyber-physical systems (CPSs) integrate programmable computing and communication capabilities to traditional physical environments. The use of SCADA (Supervisory Control And Data Acquisition) technologies to build such a new generation of CPSs plays an important role in current critical national-wide infrastructures. SCADA-driven CPSs can be disrupted by cyber-physical attacks, putting at risk human safety, environmental regulation and industrial work. In this paper, we address the aforementioned issues and provide a discussion on the mitigation techniques that aim to optimize the recovery response when a SCADA-driven CPS is under attack. Our discussion paves the way for novel cyber resilience techniques, focusing on the programmable computing and communication capabilities of CPSs, towards new research directions to tolerate cyber-physical attacks.

1 Introduction

Current Cyber-Physical Systems (CPSs) integrate modern computation and networking resources to control physical processes. These systems use sensor measurements to get information about physical processes, then control processing units to analyze and make decisions that are performed by system actuators, e.g., to maintain the stability of the physical processes. Supervisory Control And Data Acquisition (SCADA) is a traditional technology to build CPSs. SCADA protocols can be used to monitor and control hardware that may be separated by relatively large distances. SCADA-driven CPSs play an important role in most national critical infrastructures. This includes electrical transmission, energy distribution, manufacturing and supply chain, waste recycling, public transportation, e-health, financial services and several others.

Disruption of SCADA-driven CPSs have a direct impact in the physical world. Cyber-physical attacks may lead to negative impact on human safety, cause harm in natural environments, interrupt industrial process continuity,

N. Zincir-Heywood et al. (Eds.): FPS 2018, LNCS 11358, pp. 1–8, 2019.
https://doi.org/10.1007/978-3-030-18419-3_17

hence leading to large economic losses, generate legal problems and damage the reputation of the affected organizations [9,12].

Traditional protection techniques have been centered in detection and prevention methodologies, in order to build preemptive approaches that aim at identifying and responding to potential threats, even when vulnerability removal is not possible [10,11]. Attack tolerance should be enforced in such environments, in order to provide a correct service even in the presence of successful attacks against the system. The resulting systems should satisfy high availability requirements to guarantee the execution of the critical tasks. To guarantee that the whole system remain operational even in the presence of attacks, the activation of mitigation techniques shall be enforced, even if that means to work under graceful degradation modes.

Graceful degradation is the ability of a system to continue functioning even after parts of the system have been damaged, compromised or destroyed. The efficiency of the system working in graceful degradation usually is lower than the normal performance and it may decrease as the number of failing components grows. The purpose is to prevent a catastrophic failure of the system. In this paper, we address the aforementioned issue, and discuss the use of mitigation techniques aiming to ease and improve recovery response when a SCADA-driven CPS is under attack. Our discussion paves the way for orchestration and configuration of novel techniques, focusing on current capabilities of modern SCADA systems, capable of enabling programmable computing and communication functionalities.

The paper is structured as follows. Section 2 provides the background. Section 3 surveys representative mitigation techniques for SCADA-driven CPSs. Section 4 provides a discussion about a novel mitigation technique and how to optimize the response of the system under an attack using the existing mitigation approaches. Section 5 provides the conclusions of the work.

2 Background

SCADA-driven CPSs have special requirements that may affect traditional security mechanisms designed for traditional IT systems. In the following, some of these requirements are reviewed.

2.1 Cyber-Physical Systems

A Cyber-Physical System (CPS) consists of two main parts. First, a programmable (cyber) layer, containing the computing and network functionalities. Second, a physical layer, representing dynamic automation processes. Both together hold the series of distributed resources leading the environment that is expected to monitor the behavior of physical phenomena, as well as to taking the necessary actions to get control over them.

The components of the cyber layer control the behavior of the physical layer and the feedback of the physical layer affects the decisions of the cyber layer.

The CPS become smarter as the interaction between physical and cyber layers grows up. As a consequence, they get more vulnerable to attacks. The integration between layers is an important point to evaluate and determine how the information flow should be protected.

The cyber layer uses security mechanisms similar to the mechanisms for traditional information systems. The physical layer has different requirements and can be controlled in different ways. For example, considering the model of the involved physical process. However, it is important to see the system as a whole, also thinking about the information flows to and from the cyber layer and the interconnected networks to determine how to protect them.

Most of the proposals work only on the cyber layer or in the physical layer in a separate way, without keeping in mind the information flows from one layer to another or the communication patterns allowed between different components. Security solutions should work on a more general approach that takes into account attack vectors that exploit vulnerabilities of different components at the same time instead of focusing on analyzing isolated components. In addition, correlating events may help to detect security incidents that occur in different components and can not be detected analyzing the components separately.

2.2 Specificity and Characteristics of SCADA-Driven Systems

SCADA-driven systems have particular characteristics and requirements that must be taken into account with respect to traditional information systems. For instance, they may need to satisfy *real time constraints*, e.g., for the execution of critical functionality. This may include satisfying high speed communications, synchronization of tasks, etc.

A second characteristic is the priority of such systems at guaranteeing *availability* constraints. Indeed, availability and service continuity requirements in SCADA-driven systems are a crucial factor, more than any other property addressed from an information technology standpoint. The whole system may require to prove and fulfill the ability of the system in order to continue its operations through enabling redundant controls, while some mitigation patches are applied to the affected resources, even if this requires moving the system to degraded state modes.

Finally, other representative characteristics may lay on the *geographic distribution of system components* (e.g., some of the monitoring or surveyed nodes being deployed at remote locations), the *constrained nature of the computation resources* associated to some of the field devices (e.g., in terms of memory and processing power), and the existence of outdated and highly vulnerable *legacy systems*, given the lifetime cycle of SCADA resources (much longer than any other equivalent technology deployed over traditional information systems).

Attacks against SCADA-driven CPSs may exploit vulnerabilities at both the physical and the cyber layers. Regardless the layer, attacks may exploit vulnerabilities in the associated resources of the whole system, such as transport protocols, low-level transmission technologies, system-oriented monitoring languages, etc.

3 Mitigation Techniques

The main objective of protecting a SCADA-driven CPS is to enable attack tolerance while satisfying the requirements listed in Sect. 2.2. Attack tolerance assumes that a system remains to a certain extent vulnerable and attacks on components can happen and be successful. However, the CPS must ensure that the overall system nevertheless remains operational and can continue to function under graceful degradation [16].

One of the earliest techniques to respond and mitigate the effect of an attack is the use of redundancy. This technique uses alternative copies of, e.g., system components, in order to guarantee system availability. If the system finds that the output values of a primary component are not correct (e.g., according with the output values of redundant components), then the responsibility is transferred to one of the redundant components assuming that there was an infiltration or compromise. This technique is mostly used for achieving fault tolerance in case one of the components fail. However, its use for security purposes has some drawbacks. Since the replicas are identical, if the attacker manages to compromise one of them, then the rest of the replicas can be compromised too. Nonetheless, if the replicas are placed geographically distributed, then this approach may be useful for attacks that exploit vulnerabilities that require physical access to the devices. But this mitigation approach would be based on the inability of the attacker to physically reach the other locations. Regardless, this would not solve the possible malicious access through the network. For this reason, redundancy is often combined with the use of diversity. The goal is to guarantee the existence of different replicas failing in an independently manner and with non-overlapping patterns. To achieve this, replicas may hold different hardware, platform or software. Thus, the system is protected from specific infrastructure failures, errors, bugs or vulnerabilities. Authors in [4] use this approach with hardware and software diversity to achieve cyber-resilience of industrial control systems. This technique may also be used to achieve resilient web services [6,7]. This approach increases the management complexity of the platform as well as the effort required to control the vulnerabilities and keep all the components correctly patched. It also increase the required investment to acquire the redundant components.

In a similar vein, recovery techniques may also help to repair the damages caused by an attack against the systems. For instance, imagine the situation in which roll-back actions can be enforced to returning the system to a previous state that was considered as correct prior an attack. If this is possible, complementary techniques could include resuming and deploying of operations, re-execution of disrupted operations, as well as re-installation of corrupted files and renewal of cryptography [16]. Other examples include the use of periodic software rejuvenation in a proactive and reactive mode [14], in a way in which a system gets recovered automatically and periodically whenever an attack against the system is detected. However, this approach has the same problem as the redundancy technique, since the vulnerabilities are not removed, the attacker may compromise the system again.

Another example is the use of *model-based responses*, in which a model of the physical process is computed in order to generate an approximation of the normal behavior of the system prior the execution of an attack. This approach can use as input parameters an estimation of the true value of, e.g., system sensors; as well as the resulting value after the attack has occurred, i.e., right after the sensor has been tampered [3]. This approach should be used only as a temporal solution because using a simulation value instead of the real sensor measured opens the control loop and this may bring cause problems in the system behavior.

Mechanisms to detect attacks in SCADA-driven CPSs have been developed but there is very little work on how to automatically respond to them. Most of the response solutions are manual or aim at absorbing the impact of the attack through redundancy, diversity, restoration or containment techniques. However, more effective solutions that take into account the dynamic and changing nature of an attacker could be achieved, e.g. using a technique that considers dynamic adaptation of the system as a reaction, i.e. the system could deploy different defense policies depending on the attack or it could also modify the executed actions as the attack is going on. In this line, authors in [8,13] have proposed response solutions based in the network reconfiguration. In these solutions, the network controller coordinates the mitigation strategies. However, this approach could be enhanced using software reflection, in order to achieve a system capable of modifying the code to achieve state reparation. Software reflection is a technique appropriate for high-level languages. However, since the SCADA-driven CPSs controllers are located within the cyber layer of the system, they can monitor the system and apply these mitigation techniques.

4 Mitigation of Attacks Using Software Reflection

A promising technique to be fully explored under the problem domain addressed in this paper deals with the use of *software reflection* to handle attack on SCADA-driven CPSs. Software reflection is a meta-programming technique that allows a system to adapt itself through the ability of examining and modifying its execution behavior at runtime. As a mitigation technique, software reflection has the potential to allow a system to react and defend itself against availability threats. When a malicious activity is detected, the system shall dynamically change the implementation to activate remediation techniques to guarantee that the system will continue to work. During the development process it is important to do non-regression testing over the alternatives defense implementation to verify that the system is still correct according to the functional specifications.

Notice that software reflection provides the ability to analyze, inspect and modify the structure and behavior of an application at runtime. This allows the code to inspect other code within the same system or even itself. Reflection allows inspecting classes, examining fields, changing accessibility flags, dynamic class loading, method invocation and attribute usage at runtime even if that information is unavailable at compile time.

This kind of approach has been successfully explored to mitigate attacks against Internet web services [2] restoring the interface of a system to the state

previous to the attack. The idea relies on enabling reflection as a remediation techniques, when attacks are detected. However, research challenges include the application of this technique in more complex scenarios as well as modeling and orchestration of the appropriate plans in order to activate the technique, while guaranteeing the availability of the service, even if offered under a degradation mode.

Software reflection can complement the list of mitigation techniques listed in Sect. 3, in order to enable an optimal management of attacks against SCADA-driven CPSs. Research work remains to be explored, in order to put in practice such a solution. Some concerns, directions and discussions to address the aforementioned goal is presented in the sequel.

4.1 Discussion and Research Directions

The first concern deals with *performance overhead*. Reflection involves programming types that are dynamically resolved. Some optimization that typically are done in advance may risk to fail performance guarantees. In addition, reflection takes execution time and memory to discover and manipulate class properties during the runtime execution of the system. Reflective operations may suffer from slower performance than their non-reflective counterparts. It should be avoided in sections of a code that are called frequently in performance sensitive applications [5]. This may be used or restricted to small code sections that are not intensively called.

The second concern is in terms of access restrictions. Reflection requires runtime permissions which may not be present when running normally. This is an important consideration for code which has to run in a restricted security context [1,5]. Reflection may allow to access and update fields, and execute methods that are forbidden by normal access or visibility rules. This is achieved due to reflection breaks object encapsulation. If this technique is not used properly, it increases considerably the attack surface of a program and may allow malicious access to information that is supposed to be hidden, access files on the local machine, allow the injection of malicious native code or load restricted classes.

Solutions to the aforementioned problem exist. For instance, the issue may be addressed by using dynamic object ownership concepts, e.g., to design an access control policy to reflective operations [15]. This policy grants objects full reflective power over the objects they own but limited reflective power over other objects. This is done through an object ownership relation to determine access rights to reflective operations on a per-object basis and based on the dynamic arrangement of objects rather than on static relations between structural entities.

In terms of complexity, reflection procedures fail more often at runtime than during compilation. For instance, changing the object to be loaded will probably cause the generated loading class to throw a compilation error, but the reflexive procedure will not see any difference until the class is used during runtime. Moreover, determining exactly what the code is doing is quite complex due to reflection can obscure what is actually going. So, the only way to truly determine its behavior is to execute the code and see how it would behave at runtime with

sample data. However, to do this for every possible data combination is nearly impossible. In addition, this increases the complexity of the maintenance of the software due to the reflection code is also more complex to understand than the corresponding direct code. Many of the security problems are due to human programming errors and this approach increases the complexity of the solution. So, it also increases the likelihood of errors in the code.

Finally, languages with software reflection capabilities must to be explored under the problem domain context of SCADA-driven CPSs. Note that reflection is an advanced development technique for high-level languages such as Java, Python and Ruby. The possibility of extending traditional languages for the family of systems explored in this paper, e.g., given the low resource capabilities of some of the components, may be a barrier to our proposal. Nevertheless, the technique may certainly be applied by those component at the cyber layers, holding much less resource constraints.

5 Conclusion

This paper has surveyed some current trends in terms of mitigation techniques aiming to optimizing the recovery response of cyber-physical systems (CPSs) under attack. We have focused on CPS built using SCADA (Supervisory Control And Data Acquisition) technologies, in order to provide their computing and communication capabilities, beyond traditional physical components. We have enumerated some ongoing solutions in order to build higher resilient environments. We argued that the use of software reflection, in addition to traditional techniques such as redundancy, diversity and automated recovery is a promising way to enable an efficient response under the presence of cyber-physical attacks. We have also discussed some concerns and limitations that would deserve new research directions to enable and orchestrate such a technique, in order to drive our next steps and future work.

Acknowledgements. The authors acknowledge support from the Cyber CNI chair of the Institut Mines-Télécom. The chair is supported by Airbus Defence and Space, Amossys, EDF, Orange, La Poste, Nokia, Société Générale and the Regional Council of Brittany. The chair has been acknowledged by the Center of excellence in Cybersecurity. The authors also acknowledge support from the European Commission, in the framework of the H2020 SPARTA project, under grant agreement 830892.

References

1. Security considerations for reflection: https://docs.microsoft.com/en-us/dotnet/framework/reflection-and-codedom/security-considerations-for-reflection. Accessed 23 Aug 2018
2. Cavalli, A.R., Ortiz, A.M., Ouffoué, G., Sanchez, C.A., Zaïdi, F.: Design of a secure shield for internet and web-based services using software reflection. In: Jin, H., Wang, Q., Zhang, L.-J. (eds.) ICWS 2018. LNCS, vol. 10966, pp. 472–486. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94289-6_30

3. Cómbita, L.F., Giraldo, J., Cárdenas, A.A., Quijano, N.: Response and reconfiguration of cyber-physical control systems: a survey. In: 2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC), pp. 1–6. IEEE (2015)
4. Kim, C.: Cyber-resilient industrial control system with diversified architecture and bus monitoring. In: 2016 World Congress on Industrial Control Systems Security (WCICSS), pp. 1–6. IEEE (2016)
5. Oracle, J.D.: The reflection API. https://docs.oracle.com/javase/tutorial/reflect/. Accessed 23 Aug 2018
6. Ouffoué, G., Zaidi, F., Cavalli, A.R., Lallali, M.: How web services can be tolerant to intruders through diversification. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 436–443. IEEE (2017)
7. Ouffoué, G., Zaidi, F., Cavalli, A.R., Lallali, M.: Model-based attack tolerance. In: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 68–73. IEEE (2017)
8. Piedrahita, A.F.M., Gaur, V., Giraldo, J., Cardenas, A.A., Rueda, S.J.: Virtual incident response functions in control systems. Comput. Netw. **135**, 147–159 (2018)
9. Rubio-Hernan, J., De Cicco, L., Garcia-Alfaro, J.: Event-triggered watermarking control to handle cyber-physical integrity attacks. In: Brumley, B.B., Röning, J. (eds.) NordSec 2016. LNCS, vol. 10014, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47560-8_1
10. Rubio-Hernan, J., De Cicco, L., Garcia-Alfaro, J.: Revisiting a watermark-based detection scheme to handle cyber-physical attacks. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 21–28. IEEE (2016)
11. Rubio-Hernan, J., De Cicco, L., Garcia-Alfaro, J.: On the use of watermark-based schemes to detect cyber-physical attacks. EURASIP J. Inf. Secur. **2017**(1), 8 (2017)
12. Rubio-Hernan, J., De Cicco, L., Garcia-Alfaro, J.: Adaptive control-theoretic detection of integrity attacks against cyber-physical industrial systems. Trans. Emerg. Telecommun. Technol. **29**(7), e3209 (2018)
13. Rubio-Hernan, J., Sahay, R., De Cicco, L., Garcia-Alfaro, J.: Cyber-physical architecture assisted by programmable networking. Internet Technol. Lett. **1**(4), 44 (2018)
14. Sousa, P., Bessani, A.N., Correia, M., Neves, N.F., Verissimo, P.: Resilient intrusion tolerance through proactive and reactive recovery. In: 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007), pp. 373–380 (2007)
15. Teruel, C., Ducasse, S., Cassou, D., Denker, M.: Access control to reflection with object ownership. In: Proceedings of the 11th Symposium on Dynamic Languages, DLS 2015, pp. 168–176. ACM, New York (2015)
16. Veríssimo, P.E., Neves, N.F., Correia, M.P.: Intrusion-tolerant architectures: concepts and design. In: de Lemos, R., Gacek, C., Romanovsky, A. (eds.) WADS 2002. LNCS, vol. 2677, pp. 3–36. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45177-3_1