Resilient Self-Reconfiguring Industrial Control Systems

Jolahn Vaudey
Gwenaël Delaval, Stéphane Mocanu, Eric Rutten
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG,
38000 Grenoble, France
Email: firstname.name@inria.fr

***Abstract*—This paper presents a Ph.D topic on reconfiguration in reaction to attacks in industrial control systems. The main idea is to redistribute tasks located on compromised machines to the remaining ones, to keep the system in a safe state. We consider using discrete controller synthesis techniques or constraint programming to generate the reconfiguration controller. We build upon the results of prior work on the subject[1]. The goal would be to design a process that takes as input a description of the system and of the properties we want to enforce, and output a corresponding controller that can run on actual industrial control hardware.**

I. Context

This Ph.D subject concerns the development of mechanisms deployed for reaction to intrusions in Industrial Control Systems (ICS). These systems consist of an interconnection of physical and logical components controlling a physical process. This is usually orchestrated by a network of Programmable Logic Controller (PLC), rugged computers commonly used in the industry. Here, we are interested in mechanisms allowing such a network to automatically reconfigure itself while under attack, in order to maintain safety properties. The traditional approach to this issue in information networks is to isolate compromised devices. However, this is poorly suited to industrial systems, for which availability is a priority: isolating a compromised controller leaves its part of the physical process unsupervised which in turn leads to unacceptable hazardous situations. Additionally, any reconfiguration action taken must be quick, as most components in an ICS have to respect strict real time constraints.

II. Objective

Acknowledging these specificities of ICS, we will be aiming at synthesizing controllers able to reconfigure the system under attack to keep control of the physical process. This controller would take as input alerts raised by Intrusion Detection Systems (IDS) [2], allowing it to identify compromised PLCs. Then, the controller's action should be twofold. On one hand, it will reconfigure the communication network by isolating compromised equipments. On the other hand it will also modify the command, by migrating the control programs that were running on the now isolated components to maintain the supervised physical process' integrity and safety properties. To enforce quick reaction times, all tasks a PLC could run should be integrated in its main loop.

A similar method has already been tested on car systems [3]. In this paper they moved programs to a specific backup controller when the system was attacked, while in our case we will try to execute them on the remaining PLCs. These controllers are not assumed to be identical, and there may be different implementations of the programs implementing the tasks to reduce the impact of specific vulnerabilities. Thus, the "value" obtained by running these programs may differ based on the machine executing them. As the processing power decreases, some of the control programs will need to disable features incompatible with their new hardware. As a last resort and only if it is possible, some tasks may need to be switched to a "safe" mode as in stopping every part of the program save for those that keep the physical system safe.

III. Prior works

This thesis will build upon the results of a prior study [1]. In this paper, a prototype of the reconfiguration process was devised, and a simulation for a generic system was written and executed in Heptagon. Here, the centralized controller managing the reconfiguration is assumed to be both never compromised and always connected to all PLCs. The main takeaway is that we can synthesize controllers using Heptagon/BZR for a simplified model of a ICS. While the controllers obtained using synthesis are maximally permissive, these tools also allows us to select configurations that maximize the number of programs running in nominal mode for all steps. Constraints such as critical sections and dependencies have also been modeled. The caveat though, is that due to exponential complexity, we cannot synthesize

controllers for large systems. In fact, the kind of system described in the article stops compiling after the number of tasks and PLCs becomes superior to 7. Thus, if we were to tackle real ICS, we would need to decompose them in smaller sub-systems to use Heptagon. A perspective described in the paper would be to use the system structure, dividing it in mostly independent physical subprocesses, and dedicating small controllers to these local loops. More global properties could be enforced with other controllers that would supervise these local ones, forming a hierarchical structure.

IV. Approach

To generate this controller, we started by using discrete controller synthesis (DCS) techniques [4]. The system can be described using Heptagon, a synchronous dataflow programming language inspired by Lustre [5]. This language allows us to model both the behaviour of the system's components and the safety properties we want to enforce. This method allowed us to create an ad-hoc controller, but complexity issues arise even for small scale systems and thus we will prefer alternative methods for the controller.

Since our goals entails generating the controller from a description, we are interested in the IEC 62443 standard. It pertains to cybersecurity topics for OT systems, and among other things defines models for ICS. In the recommended architecture, plants are divided in zones, grouping components that shares security needs.

As no solution seemed to exist to formally describe such a system, and as the standard provides no example, we had to design a DSL that can describe instances of IEC 62443 compliant systems. This was done in the Eclipse suite, using Xtext to generate the grammar and the textual IDE, and Sirius for the graphical IDE.

To create a controller from this description, we need additional information, as the standard apply to static systems configuration, while it can show where a given program is executed, it does not show where it could be moved to. A property field was added to devices and applications, which also got a new requirement field. These requirements are boolean expression that can take into account the application's properties, or those of the device it runs on. To be executed on a given hardware, the program's mandatory requirements must all be fulfilled. A score expression was added, also taking into account those properties and giving the abstract "value" obtained from the application running on a particular device. A stopped program has a score of 0 by definition.

Using this additional information, the controller is created as such: whenever a device is compromised a constraint program is generated. Every application-device association that contradicts a requirement, or that does not allow a needed communication is forbidden, and thus the constraint program is designed so every solution is a valid configuration, using only non-attacked devices. In part to avoid choosing trivial solutions (stopping every applications for instance), the constraint program also maximize the total score across applications. Finally, the controller can output either the first valid solution that maximizes the score, or all such configurations leaving the final choice to users. This second option is much slower, but since the system, in this scenario, will have to wait for an operator's input, this is not a problem.

To ensure that this method could work in real conditions, the synthesized controllers will be implemented to run and be tested on a physical PLC-based platform. Currently, we use a FischerTechnik learning factory controlled by Schneider PLCs.

V. Conclusion

In this paper, we presented a Ph.D topic, in which we will try to develop a method to synthesise a controller able to reconfigure an industrial control system under attack. This method should output a program able to control a real physical process by manipulating its PLC, and increase the system's availability.

References

[1] G. Delaval, A. Hore, S. Mocanu, L. Muller, and Éric Rutten, "Discrete control of response for cybersecurity in industrial control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1747–1754, 2020, 21st IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320329554

[2] H. Debar, M. Dacier, and A. Wespi, "Revised taxonomy for intrusion-detection systems," *Annales des Télécommunications*, vol. 55, pp. 361–378, 07 2000.

[3] B. Potteiger, Z. Zhang, and X. Koutsoukos, "Integrated instruction set randomization and control reconfiguration for securing cyber-physical systems," in *Proc. of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security*, ser. HoTSoS '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190619.3190636

[4] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proc. of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[5] G. Delaval, E. Rutten, and H. Marchand, "Integrating Discrete Controller Synthesis into a Reactive Programming Language Compiler," *Discrete Event Dynamic Systems*, vol. 23, no. 4, pp. 385–418, 2013. [Online]. Available: https://hal.inria.fr/hal-00863286