Work-in-Progress: The Cyber-Physical Immune System

Bo Pang*, Ashank Verma*, Jingchao Zhou*, Inigo Incer, Alberto Sangiovanni-Vincentelli
{pangb,ashankv,jingchao_zhou,inigo,alberto}@berkeley.edu
University of California, Berkeley

* Equal contribution.

ABSTRACT

Cyber-Physical Systems (CPS) are important components of critical infrastructure and must operate with high levels of reliability and security. We propose a conceptual approach to securing CPSs: the Cyber-Physical Immune System (CPIS), a collection of hardware and software elements deployed on top of a conventional CPS. Inspired by its biological counterpart, the CPIS comprises an independent network of distributed computing units that collects data from the conventional CPS, utilizes data-driven techniques to identify threats, adapts to the changing environment, alerts the user of any threats or anomalies, and deploys threat-mitigation strategies.

1 INTRODUCTION

A Cyber-Physical System (CPS) comprises distributed and interconnected computing units that interact with the physical environment. CPSs thus offer a vast surface area for attackers to exploit [2]. Attackers could execute memory corruption attacks in the control software or spoof the perception data gathered by the system, such as data from the speed sensors of a vehicle or from the voltage sensors in a power station [5].

CPSs with fixed physical boundaries and constituents, such as vehicles or airplanes, are akin to biological organisms, which are often composed of a fixed set of constituents. We derive inspiration from biology in order to provide our CPSs with an immune system in the same way in which biological organisms have one. We know that the immune system of mammals is composed of elements that constantly monitor the body for unusual behavior and alert the central immune system of suspicious activity, and of elements that respond to threats. Some of the body's responses are innate, and some are learned from the experience of fighting a threat [6].

Similarly, the CPIS comprises an independent network of distributed computing units that attach onto the existing CPS. CPIS Monitors collect execution traces from the computing units present in the conventional CPS and potentially respond to threats upon detection, as shown in Figure 1. The immune system also contains the CPIS Main Computing Unit, which collects data from all monitors and detects anomalies with its system-wide point of view.

The CPIS has both innate and adaptive immunity, similar to the biological immune system. Specifically, the CPIS displays innate immunity by quickly finding known issues or threats. It establishes adaptive immunity by monitoring the CPS and identifying anomalies using data-driven techniques, for which the CPIS is not pre-programmed.

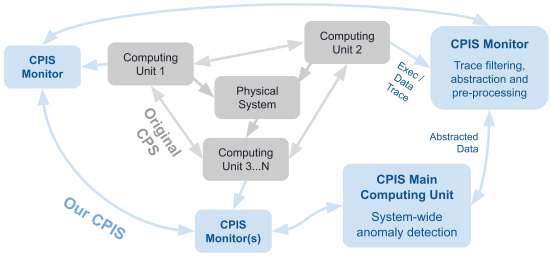

Figure 1: *CPIS Architecture*

Since the CPIS is a conceptual approach to securing a CPS, its deployment could use specific techniques for attack detection and mitigation that have been considered in the literature. Giraldo et al. [4] survey threat detection techniques for CPSs that make use of knowledge of the physics of the system and its environment. An appeal to "physical coherence" is, in fact, a security tool that is available in CPSs, but absent from the traditional security toolbox. Combita et al. [3] survey some responses that a CPS could conceivably deploy when an attack is detected. The notion of developing a security system based on inspiration from biology has precedents. The closest in spirit to our approach is an architecture by Roman et al. [7], which uses virtual machines that traverse an IoT system to monitor and defend it. Our approach is complementary in the sense that we propose to leave the original CPS intact, and ensure security using additional hardware and software whose exclusive purpose is the implementation of the CPIS. Scadman, a recently-proposed architecture by Adepu et al. [1], secures a SCADA system by using a central node which has access to sensor and actuator values as reported by PLCs used in the system. The CPIS differs from this notion because it monitors the execution traces of every computing node in the system, allowing it to detect cyber attacks to any processing element.

2 AN IMPLEMENTATION OF THE CPIS

To validate the CPIS concept, we implemented an illustrative vehicular cruise control CPS using Mininet[1]. The CPS consists of an engine controller, a cruise controller, and a simulated physical environment (shown in the LHS of Figure 2). The cruise controller runs a PID algorithm to determine desired vehicle acceleration, and sends it to the engine controller via TCP; the engine controller determines the gear and throttle accordingly. The physical environment is driven by a simulator that calculates vehicle acceleration and

[1] http://mininet.org/

EMSOFT'21 Companion, October 8–15, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8712-5/21/10.
https://doi.org/10.1145/3477244.3477621

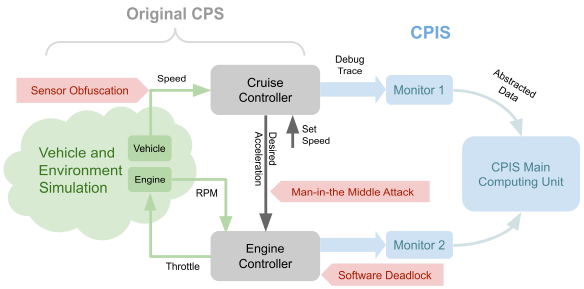

Figure 2: *CPIS prototype attached to a cruise control CPS*

speed, taking into account the vehicle mass, road incline, engine torque curve, rolling friction, and air resistance. In this CPS, the controllers represent the two computing units that jointly provide throttle and gear inputs to the simulated physical environment, and receive sensor data (speed and engine RPM) from the environment.

We then implemented a CPIS that attaches to the vehicular CPS. The CPIS consists of two monitors that peer with the two CPS computing units, and a CPIS main computing unit that connects to both monitors (shown in the RHS of Figure 2). Each monitor performs two major tasks. The first of these tasks is to obtain debug traces from its peer CPS computing unit and abstract these traces. The incoming debug traces consist of execution traces indicating the line of code being executed by the CPS computing unit, and data traces showing the change of the stack variables. The execution traces are merged into a line-counting vector (each element in the vector represents the amount of times the corresponding line has been executed). The data traces are filtered by regular expressions and used to update a local copy in the monitor. The second task of the monitor is to pre-process the abstracted trace information using a pre-programmed set of rules. For example, the data copy representing vehicle speed and engine RPM should never exceed physical limits; the counters for lines of repetitive executable code should keep incrementing. The pre-processing of information allows each monitor to quickly identify and report obvious issues, mimicking innate immunity.

The CPIS main computing unit periodically collects and further processes information abstracted by both monitors, giving it a global view of the CPS. It consolidates copies of run-time data from both monitors, including vehicle speed, gear number and throttle value. Without built-in knowledge of how these data are correlated, the main computing unit periodically trains a regression model to learn the relationships between these heterogeneous data inputs. For example, after driving the vehicle using a variety of speed settings, the CPIS prototype is able to successfully capture the correlation between throttle percentage, the reciprocal of gear number, the second power of speed, and vehicle acceleration. It then uses the trained model to validate newly collected data, and report anomalies if the error exceeds a predetermined margin repeatedly.

The main computing unit also merges line-counting vectors from both monitors into a single vector, from which it finds linear invariants within the counter values belonging to both CPS computing units. For example, the code in the cruise controller that sends data should always execute in sync with the code in the engine controller that receives the data, yielding $1:1$ correlation on their counter values representing the send & receive code stack. Such linear invariants are used to validate newly obtained counting vectors and help determine system anomalies.

To exercise the CPIS prototype, we simulate three run time anomalies in the CPS, shown as red pointers in Figure 2. The first is sensor obfuscation, implemented by randomly overwriting speed data provided by the simulator. The second is a man-in-the-middle network attack. We implement an attacker script that hijacks and randomly modifies the *desired acceleration* data sent from the cruise controller to the engine controller. The third is software deadlock, implemented by starving the software execution in the engine controller and causing it to halt. We repeatedly introduce one of these anomalies, and the CPIS prototype is able to identify over 85% of them within seconds, while making zero false-positive judgements.

3 CONCLUSION AND FUTURE WORK

We conclude that the CPIS prototype shows the effectiveness of combining localized and system-wide anomaly detection via CPIS monitors and the main computing unit. The CPIS monitors help detect anomalies quickly using their pre-programmed information, especially during sensor obfuscation; whereas the CPIS main computing unit plays an important role during man-in-the-middle attacks by cross-validating run-time data that are seemingly valid to individual monitors.

As part of our immediate goals for the CPIS, we plan to improve the trace-summarization capabilities of the supervisory nodes. Since each monitor will be exposed to large amounts of trace data, especially on a high-speed multi-core computing unit, deciding what to keep and communicate to the main node is an important problem. Another area of future work is extending the capabilities of the CPIS. In our example, the CPIS detects malicious behaviors and alerts operators. We intend to add threat-mitigation capabilities, so the CPIS can automatically take countermeasures, just like the immune system does.

REFERENCES

[1] Sridhar Adepu, Ferdinand Brasser, Luis Garcia, Michael Rodler, Lucas Davi, Ahmad-Reza Sadeghi, and Saman A. Zonouz. 2018. Control Behavior Integrity for Distributed Cyber-Physical Systems. *CoRR* abs/1812.08310 (2018), 15. arXiv:1812.08310 http://arxiv.org/abs/1812.08310

[2] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. USENIX Association, USA, 6.

[3] Luis F. Combita, Jairo Giraldo, Alvaro A. Cardenas, and Nicanor Quijano. 2015. Response and reconfiguration of cyber-physical control systems: A survey. In *2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC)*. IEEE, 1–6. https://doi.org/10.1109/ccac.2015.7345181

[4] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. 2018. A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. *ACM Comput. Surv.* 51, 4, Article 76 (July 2018), 36 pages. https://doi.org/10.1145/3203245

[5] Antonio Lima, Francisco Rocha, Marcus Völp, and Paulo Esteves-Veríssimo. 2016. Towards Safe and Secure Autonomous and Cooperative Vehicle Ecosystems. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy - CPS-SPC '16*. ACM Press, 59–70. https://doi.org/10.1145/2994487.2994489

[6] Jean S. Marshall, Richard Warrington, Wade Watson, and Harold L. Kim. 2018. An introduction to immunology and immunopathology. *Allergy, Asthma & Clinical Immunology* 14, S2 (Sept. 2018), 10. https://doi.org/10.1186/s13223-018-0278-1

[7] Rodrigo Roman, Ruben Rios, Jose A. Onieva, and Javier Lopez. 2019. Immune System for the Internet of Things Using Edge Technologies. *IEEE Internet of Things Journal* 6, 3 (2019), 4774–4781. https://doi.org/10.1109/JIOT.2018.2867613