Reflective Attenuation of Cyber-Physical Attacks

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

[1] Institut Polytechnique de Paris, CNRS SAMOVAR, Telecom SudParis, Palaiseau, France
{segovia,ana.cavalli,rubio_h,garcia_a}@telecom-sudparis.eu
[2] IMT Atlantique, Cesson Sévigné, France
nora.cuppens@imt-atlantique.fr

Abstract. Cyber-physical systems (CPS) integrate computation and networking resources to control a physical process. The adoption of new communication capabilities comes at the cost of introducing new security threats that need to be handled properly. Threats must be addressed at cyber and physical domains at the same time in order to detect and automatically mitigate the threats. In this paper, we elaborate an approach to attenuate cyber-physical attacks driven by reflective programmable networking actions, in order to take control of adversarial actions against cyber-physical systems. The approach builds upon the concept of *programmable reflection* and *programmable networking*. We validate the approach using experimental work.

Keywords: Cyber-physical security · Critical infrastructures · Attack detection · Adversary model · Attack mitigation · Networked control systems

1 Introduction

Cyber-physical systems (CPS) are modern control systems used to manage and control critical infrastructures [7]. Physical properties of such infrastructures are modeled via control-theoretic tools, e.g., *control-loops* and *feedback controllers* [16]. Feedback controllers have to be able to manage the behavior of the CPS, by confirming that the commands are executed correctly and the information coming from the physical states is consistent with the predicted behavior [26]. Feedback controllers are also used to compute corrective actions, e.g., by minimizing the deviation between a reference signal and the system output measurements.

A CPS is composed of three main layers: (1) the *physical layer*, which involves the physical process (monitored and controlled by physical sensors and physical actuators); (2) the *control layer*, which is in charge of regulating the operation of the physical process via control commands; and (3) the *cyber layer*, which

S. Katsikas et al. (Eds.): CyberICPS/SECPRE/SPOSE/ADIoT 2019, LNCS 11980, pp. 1–16, 2020.
https://doi.org/10.1007/978-3-030-42048-2_2

is responsible for monitoring operation and supervision tasks. These three layers are interconnected using a communication network. In other words, the CPS can be modeled as a networked-control system [21]. The interconnection between information and operational systems leads to new security threats [20,21]. Traditional *cyber attacks* are well know and countermeasures have been studied. However, launching a *cyber-physical attack* requires a different knowledge from the one used in traditional cyber security and different protection techniques are also required.

A cyber-physical attack causes tangible damage to physical components, for instance, adding disturbances to a physical process via exploitation of vulnerabilities in computing and networking resources of the systems (i.e., the components at the cyber layer). However, to achieve just a cyber attack, the adversary may be able to inject any input in the system but this does not necessarily mean to be able to influence the processes in the physical world. The processes and their dynamics have to be properly understood to cause a real damage [12].

A physical process has automatic safety measures and operational constraints, e.g., to disable a physical process when certain dangerous conditions are met. For instance, to properly react when a physical component fails. For this reason, an adversary who aims at damaging the physical process needs to understand how the dynamics of the physical plant works. This means that compromising and disrupting a device or communication channel used to sense or control a physical system is a necessary requirement to perform cyber-physical attacks. However, the damage can be limited if the adversary succeeds at affecting the cyber layer, but remains unable to manipulating the control system (i.e., fails at perturbing the physical process). To achieve the desired impact and achieve a cyber-physical attack, the adversary needs to assess how the attack will perform at the control level. Therefore, to achieve a cyber-physical attack, the first step is to hack the cyber layer, to obtain a remote access within the target system. Then, the second step is to learn about the physical process and how the control layer works in order to manipulate the physical layer and cause a damage to physical components. Adversaries need to know how the physical process is controlled, failure conditions of the equipment, process behavior and signal processing [20,21].

In this paper, we propose a technique to attenuate cyber physical attacks that uses programmable reflection and programmable networks to sanitize the malicious actions introduced by some cyber-physical injection attack such as false data injection, bias injection, replay attack, command injection and cover attack [27]. The adversary uses the network to manipulate the process through the modification of specific payloads. Then, the proposed technique uses the network to neutralize the attack effects. This concept relies on the use of programmable reflection, which is a meta programming technique that has the potential to allow a programmable system manipulate itself at runtime and the use of programmable networks to sanitize the traffic.

The main contributions of the paper are summarized as follows: (1) we propose a technique to handle cyber-physical injection attacks; (2) we revisit the

use of programmable networking in [23], to achieve as well reflective attenuation of CPS attacks; and (3) we provide experimental work to validate the approach.

Paper organization—Section 2 provides the related work. Section 3 provides preliminaries and assumptions. Section 4 presents our attenuation approach. Section 5 reports the experimental work. Section 6 concludes the paper.

2 Related Work

We survey next some related work, structured in terms of *attack tolerance*, *programmable networking* and *programmable reflection*.

2.1 Attack Tolerance

A defense in depth strategy proposes to defend a system against any particular attack using several independent methods [13]. Many proposed security solutions for CPS focus in detection and attack prevention. However, preventing every single possible attack is hard to achieve and although the efforts, attacks on the systems can happen and be successful.

Attack tolerance is the capability of a system to continue functioning properly with minimal degradation of performance, despite the presence of attacks. The main techniques proposed in the literature to achieve this are [1]: indirection, voting, redundancy, diversity, dynamic reconfiguration, distributed trust, recovery and moving target defense mechanisms.

- *Indirection* separates components using an additional layer that works as a protection barrier. For instance, proxies, wrappers, virtualization and sandboxes are used in this technique.
- *Voting* can resolve differences in redundant responses, to reach consensus w.r.t. the responses of perceived non-faulty components. The process involves comparing the redundant responses and reaching an agreement on the results to find the appropriate response. It masks the attacks, thus tolerating them and providing integrity of the data.
- *Redundancy* uses extra reserved resources allocated to a system that are beyond its need in normal working conditions. If the system finds that the output values of a primary component are not correct, then the responsibility is transferred to one of the redundant components.
- *Diversity* means that a component should be implemented in multiple different ways in order to ensure independent failures with non-overlapping patterns. To achieve this, replicas shall hold diversity in terms of hardware and software. For instance, to generate software diversity it is possible to generate diverse functionality from the same source code or automatically change the configuration of a system from time to time to confuse the adversary [9].
- A *dynamic reconfiguration* takes place after the detection of an attack. In traditional systems, reconfiguration is mostly reactive and generally performed manually by the administrator. Thus, it involves some downtime. Survivable systems need an adaptive reconfiguration to be proactive, instead.

- *Distributed trust* relies on control-sharing strategies, e.g., dividing control into shares, such that the system needs to reach a given threshold prior granting control. Below the threshold, information gets concealed to the eyes of the adversary.
- *Recovery* involves detecting the attack and modifying the system to a state that ensures the correct provision of the functions.
- *Moving target defense* enables to deploy diverse mechanisms that change over time to increase the complexity, the attack surface or the cost for an attacker in order to limit the exposure of vulnerabilities and increase system resiliency.

Our proposal combines dynamic reconfiguration and recovery approaches, via control theory and programmable networked reflection techniques. It allows a CPS to reconfigure itself and neutralize the adversary actions via dynamic traffic sanitization.

2.2 Programmable Networking

Programmable networking facilitates network management. It enables efficient network configuration that can be used for neutralizing attacks. New networking functionality can be programmed using a minimal set of APIs (Application Programming Interfaces) to compose high-level services. This idea was proposed as a way to facilitate the network evolution. Some solutions such as Open Signaling [2], Active Networking [28], and Netconf [6], among others, are early programmable networking efforts and precursors to current technologies such as Software Defined Networking (SDN) [11]. In particular, SDN is a programmable networking paradigm in which the forwarding hardware is decoupled from control decisions. SDN proposes three different functionality planes: (1) data plane, (2) control plane and (3) management plane.

The data plane corresponds to the networking devices, which are responsible for forwarding the data. The control plane represents the protocols used to manage the data plane, such as, to populate the forwarding tables of the network devices. Finally, the management plane includes the high-level services and tools, used to remotely monitor and configure the control functionality. Security aspects may have an impact on different plans. For example, a network policy is defined in the management plane, then the control plane enforces the policy and the data plane executes it by forwarding data accordingly.

The idea of using programmable networks for improving security is not new. Some examples include its use for conducting DoS (Denial of Service) attack mitigation [24] and segmentation of malicious traffic [8,17,23]. Programmable networks provide a higher global visibility of the system, which is favorable for attack detection. In addition, a centralized control plane may allow further possibilities to achieve dynamic reconfiguration of network properties, e.g., application of countermeasures.

2.3 Programmable Reflection

Programmable reflection allows a system to adapt itself through the ability of examining and modifying its execution behavior at runtime. Authors in [4,10] proposed to implement programmable networks using reflective middleware platforms. These reflective middleware platforms use reflection in order to configure and adapt in runtime nonfunctional properties like timeliness, resourcing, among others. To achieve this, the architecture is based in different components that may be loaded and unloaded dynamically in order to change the behavior of the platform and structure the programmable network.

As a mitigation technique, programmable reflection has the potential to allow a system to react and defend itself against threats. When a malicious activity is detected, the system can dynamically change the implementation to activate the mitigation techniques to guarantee that the system will continue to work [25]. This kind of approaches has been explored to mitigate attacks against Internet web services [3] using reflection to restore an interface of the system that had been modified in an attack.

3 Preliminaries

We provide in this section some initial preliminaries about our assumptions in terms of system and adversarial models.

3.1 System Model

We assume a system that is already protected from a cyber security point of view. This means that the system has been created considering all the required security mechanisms according to a risk analysis of the system. However, past experience has shown that despite all the prevention actions, attacks are still possible. The proposed approach aims at improving the system resilience and protecting it in a contingency mode after the other security mechanisms failed. This way, major failures in the physical process may be prevented.

We also assume that the CPS is governed by feedback controllers applying, e.g., a closed loop model. As shown in Fig. 1(a), the feedback controller collects the *sensor* measurements y_k to determine the state of the system process. Then, the *feedback controller* determines a control input using the received data and the reference obtained from the model. Finally, it sends a control input u_k to the *plant* so that the *actuators* perform the required actions in the physical process. After this, the sensor obtains new measurements y_k and the process is repeated. The values y_k and u_k are exchanged between the feedback controller and the plant through a network. It means that the data will be forwarded through a set of network forwarding devices to reach the appropriate destination. We assume that this network is a highly distributed, with real time traffic and a dynamic system interconnected using a programmable network that is controller by a *network controller* (e.g., an SDN controller [11]). In addition, we assume a k-resilient network, where k is the number of independent paths that interconnect two nodes.

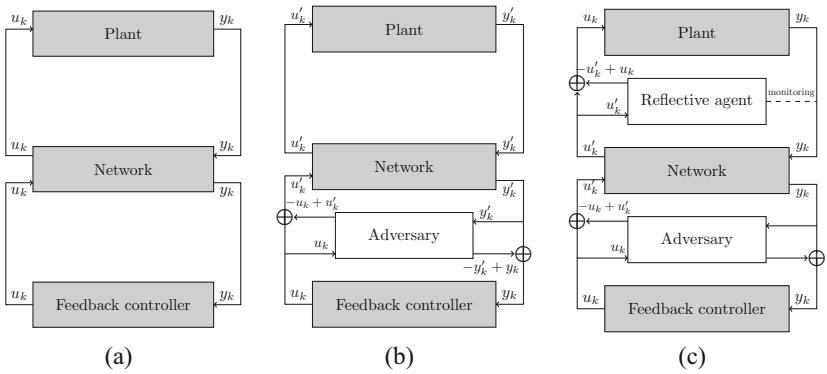


Fig. 1. Feedback control view. (a) Normal operation mode. (b) System under attack. (c) Attack attenuation.

3.2 Adversary Model

Cyber-physical adversaries have different knowledge levels about the system behavior. That knowledge is the main resource in order to build complex attacks and it may even let an adversary remain undetected if the injected data is compatible with the system dynamics. To achieve this, the adversary will try to estimate the model of the physical system and obtain its parameters. Authors in [27] propose a taxonomy of cyber-physical attacks based on the knowledge of the adversaries. Our proposal addresses the cyber-physical injection attacks mentioned in that taxonomy and we assume an insider adversary. To do this, the adversary firstly exploits a cyber vulnerability to gain access in the network channel and be able to insert or modify packets at will. After that, the physical attack to control the physical process starts. To achieve this, the adversary injects a bias in the payload of the packets containing the commands or the measures to manipulate the process. The choice of the introduced modifications depend on the specific impact the adversary wants to produce on the process.

Figure 1(b) depicts an attack against the closed-control loop. We use the traditional representation of a networked-control system. It shows the way how an adversary conducting a cyber-physical attack is represented by the control system community through block diagrams. The $\bigoplus$ symbol in the figure represents a *summing junction*, i.e., a linear element that outputs the sum of a number of input signals. As shown in Fig. 1(b), the adversary modifies the control input u_k to inject a modified u'_k value and affects the system state to disrupt normal operation conditions. Then, the adversary modifies the plant measurements y'_k to send a value y_k to the controller. This way, the controller receives a value y_k that is correlated with the command u_k that it previously sent to the plant. This can be achieved by recording and replicating previous measurements corresponding to normal operation conditions or by injecting some values calculated from the adversary estimated control model of the system.

We also assume that the adversary performs its malicious actions in the cyber-physical system, i.e., at the data layer of the network domain. This means that the adversary is not attacking the programmable network itself, e.g, the control layer. We focus on adversaries that use the network to damage the system. Adversaries that may compromise the physical nodes themselves, to damage the system, are out of the scope of this paper due to this kind of systems usually have good physical protection mechanism implemented.

4 Our Approach

Our proposed approach triggers an attack attenuation for cyber-physical attacks (i.e., disruptive attacks leading to system failures) to remain operational and provide system functionality. We assume a resilient system, capable of reacting and defending itself against known threats. Remediation starts right after attacks are detected. The system dynamically and autonomously changes its behavior to activate an attenuation plan that guarantees work continuation. This is carried out through the cooperation of the feedback controller and the network controller. Although these two controllers have different individual objectives and functionalities, they can work in a coordinated way in order to reach a common goal. Both controllers get connected and coordinate the resilience strategies, e.g., to maintain the resilient properties of the system under failure and attacks.

The proposed resilience strategies try to revert the adversary activity. This is done due to a system capable of modifying its configuration to introduce a new virtual component on-the-fly and dynamically reverting the adversary actions. The solution combines a feedback control technique to detect the attack, programmable reflection for creating the new virtual component that will help the affected feedback controller to bypass the attack and a programmable network in order to neutralize the adversary and sanitize the traffic. The complete process is composed of three main phases: (1) detection, (2) reflection and (3) traffic sanitization.

- **Phase 1 – Detection.** A feedback control detection mechanism is executed in the feedback controller. When an attack is detected, it alerts the network controller to start the coordination of the different components in the system. The physical process in a CPS can be described using an accurate mathematical model that allows to control it and detect deviations from the normal behavior. In addition, it can provide mechanisms to provide attack detection, for example using a physical watermarking that allows to authenticate the correct operation of a control system using a challenge-response detector. In our solution, we used the approach explained in [20,21]. To authenticate the exchanged information, this solution injects a known noise in the physical system signals. It is expected that the effect of that noise is also present in the measured output due to the dynamics of the system. The added noise increases the difficulty associated to the learning process of the adversary. It becomes harder for the adversary to identify the system parameters, hence

decreasing the chances of the adversary to correlate the proper input and output values.
- **Phase 2 – Reflection.** The feedback controller creates a reflective agent, which gets executed within the domain of the network controller. The reflective agent has the control capabilities associated to the victims of the attack. It uses programmable reflection to create, at runtime, a component that executes the same program and equivalent interfaces as the feedback controller. By programmable networking reflection, we refer to the system capability of modifying its networking behavior, i.e., changing accordingly to what is required. For this reason, an on-demand process for loading and unloading components as services could be performed.
- **Phase 3 – Traffic sanitization.** The forwarding elements using network programming capabilities allow to perform a dynamic network traffic sanitizing by modifying the packet containing malicious payloads. The packet affected by the adversary gets sanitized by the reflective agent, which determines what is the correct payload the packet should have. All the network actions required to sanitize the traffic are coordinated by the network controller.

The solution dynamically applies a attenuation technique using the forwarding devices to modify the traffic in order to revert the adversary actions. In a cyber-physical bias injection attack, the physical damage in the system occurs due to modified control commands injected in the plant actions. For this reason, after the traffic is modified by the adversary, the forwarding devices under the command of the network controller intercept those packets and modify them using the reflective agent that knows the physical model of the system and has the ability to determine whether those values in the packets are correct or not. In order to perform the calculations, the reflective agent uses as input the sensor measurements that the plant communicated to the feedback controller previously, since this component executes the same transmission function as the feedback controller, it can determine the correct command values without any model of normal behaviour or historic data of the system. In addition, this node can monitor the measures values, since it is in the network control level and has the potential power to see all what is happening in the network. Moreover, since the network is k-resilient this node can be placed in the most convenient path between the plant and the feedback controller.

Figure 1(c) shows how our attenuation process works in order to handle the attack perpetrated by the adversary. The adversary modifies the command u_k sent from the feedback controller to the plant in order to insert a fake command u'_k. After this, the traffic is modified again to sanitize it with the help of the reflective agent calculations that take as input the monitored sensor values y_k captured from the network. This way, the plant receives the correct u_k command and the physical process is not affected by the adversary actions. When the attack is finished, the normal operation of the system can be restored. The original controller can take over again.

To achieve this solution, the feedback controller is made of two subcomponents: (a) the control component that is in charge of enforcing the dynamical control objectives (fast dynamics are involved); (b) the supervisory component that communicates in a bi-directional way with the network controller. At the data layer of the network domain, we have network probes and effectors, conducting data monitoring—if instructed by the control domain. Network probes monitor the traffic in the data domain and provide the information to the network controller.

The network controller, based on measurements provided by network probes and feedback provided by the feedback controller, is able to detect a possible threat acting on the control path. In response to such a threat, the reflective agent provides a corrective measure to attenuate the impact. The network controller can be seen as a computing entity that is located at an external location (e.g., a kind of Network Operating System [11]). For instance, it provides resources and abstractions to manage the system using a centralized or a decentralized model [20].

Together, both controllers manage the data domain. The feedback controller manages the physical system through physical sensors and actuators deployed at the physical layer. The network controller estimates and manages the data domain through probes and effectors –deployed at the management and control domain.

The network controller analyses the information and forwards control actions to the effectors. Network rules at the control domain are responsible for enforcing such actions. For instance, when a network probe finds tampered traffic in a network path, it provides the tampered information to the control domain. Then, the network controller, located at the control domain, checks for the available resources and helps in order to enforce the action.

5 Experimental Results

We present in this section some experimental results to validate our approach. We use a physical SCADA testbed, for the generation of Modbus-driven CPS data [15]. The testbed consists of *Lego Mindstorms* EV3 bricks [19] and Raspberry Pi [14] boards that control some representative sensors (e.g., distance sensors) and actuators (e.g., dynamic speed accelerators). A sample picture of the testbed is shown in Fig. 2(a). The Modbus SCADA protocol used in the testbed is based on standard Modbus protocol specifications [15] implemented in Java. The testbed implements a kinetic dynamics use case, in which two motion devices perform a deterministic path based on linear motion (backward and forward motion over a bounded square area). We refer the reader to http://j.mp/omnetcps and [22], for additional information and video captures about this testbed. Figure 2(b) depicts a numeric co-simulation complementing the same scenario, using the collected SCADA data to train a CPS programmable simulator. The implementation uses OMNeT++ (Objective Modular Network Testbed in C++) [29,31] and leverages a series of shared APIs (Application

Programming Interfaces) over the INET [30] and SCADASim [18] libraries, to enforce the use of the Modbus protocol over TCP and UDP traffic. All the components (both in the Lego SCADA testbed and the OMNeT++ co-simulation) are synchronized by feedback controllers. Every motion device has a distance sensor in the frontal part, to measure its relative distance to the boundaries of a unit square area. The distance is transmitted to the feedback controllers via Modbus SCADA messages. The feedback controller computes the relative velocity of each motion device, and the Euclidean distance between the two motion devices, in order to guarantee spatial collision-free operations.

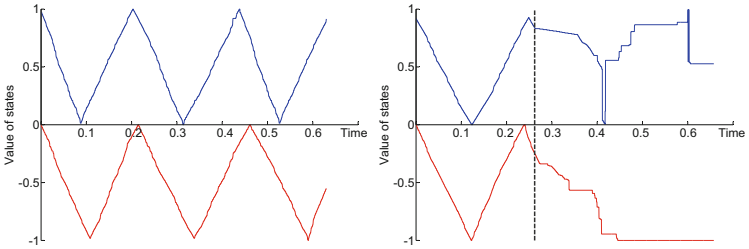
The goal of the adversary is to launch an attack at the control level to move the physical process to an undesirable state resulting in the physical collision of the two motion devices. Figure 3(a–b) show the kinetic dynamics of the system during the nominal case (i.e., absence of attacks, left-side); and during the attack (i.e., the moment at which the adversary takes control over the system, right-side). Time is normalized between 0.0 and 1.0, representing the temporal percentage of multiple experimental runs. We can appreciate how the system moves to unstable states, disrupted by the adversary. Some live demonstration videos showing the spatial collision that cause the disruption represented in Fig. 3 are available at http://j.mp/legoscada.

(a) Lego Mindstorms testbed for the generation of SCADA data.

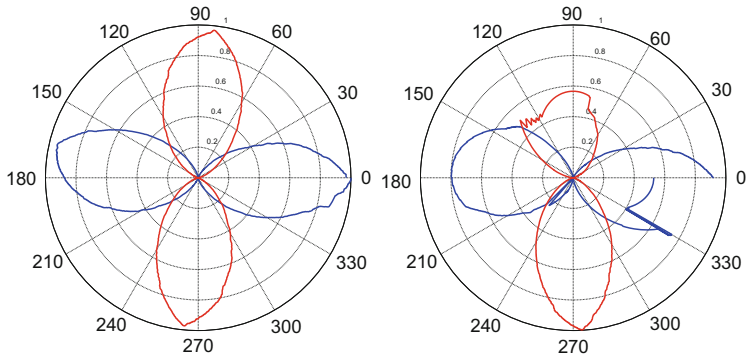
(b) OMNeT++ CPS co-simulation, using the generated data.

Fig. 2. (a) Lego testbed for the generation of SCADA-driven CPS data (cf. http://j.mp/legoscada for live demonstration videos and source code). (b) CPS co-simulation implemented over OMNeT++ (cf. http://j.mp/omnetcps for live demonstration videocaptures and source code).

During the OMNeT++ co-simulation, we analyze the system behavior in the normal operation mode, under attack and using the proposed attenuation approach. In the testbed, the two motion devices follow a trajectory of two meters. The feedback controller coordinates the movement of the motion devices, by sending the relative velocity to the motion device, and receiving back the distance of the motion device to the spatial boundaries. The feedback controller sends a series of Modbus messages to the physical environment of the plant, through a network of traffic programmable forwarders (e.g., SDN switches). The plant contains the physical process itself, the distance sensors and the actuators



(a) Temporal representation of the CPS kinetic dynamics, associated to the two motion devices (left-side, nominal mode dynamics; right-side, dynamics during the attack).



(b) Winding graph of a complete dynamics cycle (left-side, nominal case; right-side, attack case.)

Fig. 3. Lego testbed results (systems dynamics during nominal and attack modes). (a) Temporal representation (the dotted line represents the moment when an attack starts). (b) Winding graph representation of the nominal and attack modes.

that perform the commands (accelerators that increase or decrease the relative velocity of the two motion devices).

The adversary starts the cyber-physical attack by either tampering the controller with fake sensor readings or modifying the control commands sent from the controller. With the OMNeT++ co-simulation, we evaluate the attenuation of the bias injection attack, i.e., by forging tampered control commands from the controller to the plant. For simplicity reasons, we focus only on the physical part of the cyber-physical attack using the network to damage the system. In other words, we assume an adversary that already found a way to hack the cyber layer and gain remote access to the system.

Each co-simulation evaluates fifty Monte Carlo different runs. In addition, according to the sensor specification, the simulation considers a possible error of up to 1 cm w.r.t. the measured distance value. We also model the network delays using the probability distribution in [5]. Figure 4(a) shows the results obtained for the nominal case (i.e., absence of attack), considering the aforementioned

possible variation. The plots depict the average Euclidean distance, with 95% confidence intervals, between the motion devices in function of time. The horizontal axis of the plots in Fig. 4(a–d) provides a normalized time between 0.0 and 1.0, representing the temporal percentage prior concluding the simulation runs. The vertical axis of the plots in Fig. 4(a–d) provides the Euclidean distance between the two motion devices, from 0 to 1400 cm. Some further evaluation details are discussed below.

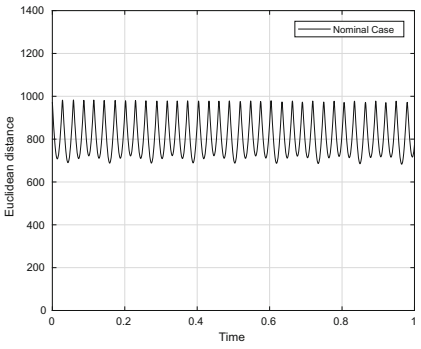
Discussion—During the perpetration of the attacks, the adversary performs a bias injection of cyber-physical data. The adversary uses the network to modify the exchanged packets between the feedback controller and the plant. We assume an adversary recording and learning the system dynamics from commands and sensor outputs. The adversary performs an initial learning phase, in order to eavesdrop data and infer the system dynamics, i.e., the same one used by the feedback controller to guarantee the stability of the system, shown as nominal case in Fig. 4(a).

Let u_k be a feedback controller command sent to the actuator of a motion device at time k. Let u_k^{act} be the command received by the actuator at time k, where $0 \leq k \leq T_s$ and T_s be the full duration of each simulation run. The attack interval T_a is limited to the simulation time T_s, as summarized next:

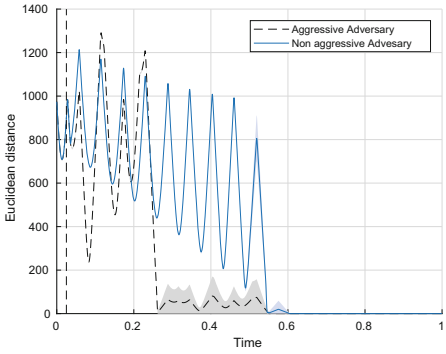
$$u_k^{act} = \begin{cases} u_k \text{ if } k \notin T_a \\ u'_k \text{ if } k \in T_a \end{cases}$$

For our evaluation, we compare two type of adversaries according to the bias injected in the payload of the packets, i.e, according to the difference between the value u'_k injected by the attacker and the real value u_k sent by the controller. This way, we define two adversary models: an *aggressive adversary* and a *non aggressive adversary*. The aggressive adversary injects in u'_k a bigger difference with respect to the correct command u_k sent by the feedback controller compared to the non aggressive adversary. In consequence, an aggressive adversary will make the system move faster from its nominal state. Figure 4(b) shows the results obtained for the two attack scenarios. The feedback controller loses its control over the system, while the adversary forces the spatial collision of the two motion devices.

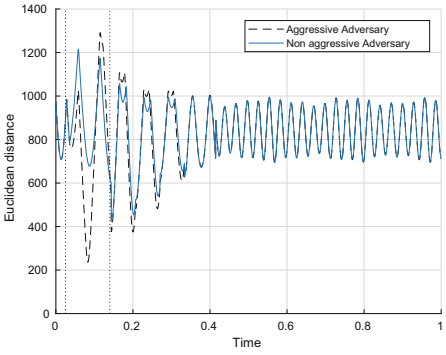
During the attenuation process, the system reacts using reflective programmable networking. The reflective agent takes control of the situation, after a hangover of the feedback controller functionality (which moves to the programmable controller domain). This reflective agent takes control over the adversary communications and neutralizes the attack. For each of the defined adversaries, we simulate two scenarios using different values for the time the solution starts working. This is a parameter of the simulation that depends mainly on the time required for the detection mechanism to detect the attack plus the time required to set up and coordinate all the components working in the approach. Figure 4(c)–(d) show how the approach guarantees the controllability property. The first vertical dotted line shows the moment when the attack starts and the second vertical dotted line shows the moment when the technique starts. It is



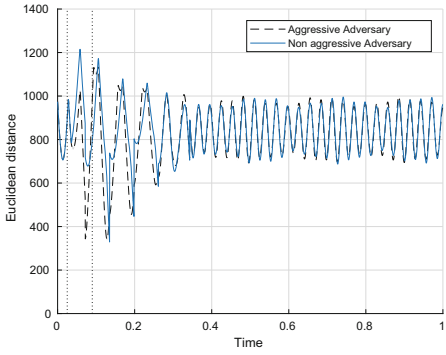
(a) Nominal Case



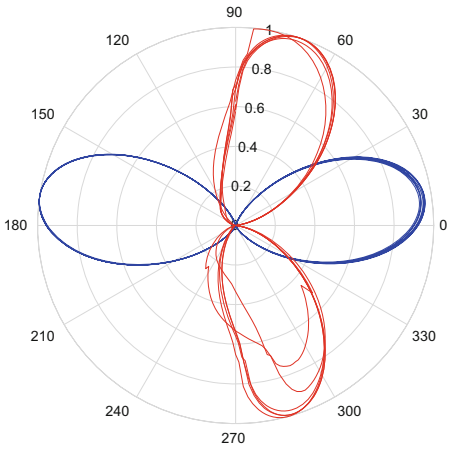
(b) Attack Case



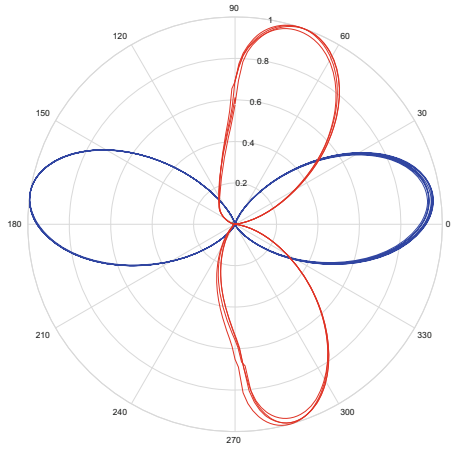
(c) Attenuation model 1



(d) Attenuation model 2



(e) Winding graph of dynamics cycles (attenuation 1)



(f) Winding graph of dynamics cycles (attenuation 2)

Fig. 4. OMNeT++ results. (a–b) Euclidean distance (with 95% confidence intervals), nominal and attack simulations. (c–d) Euclidean distance, attenuation of two different remediation starting time models. (e–f) Winding graphs, same attenuation models.

possible to appreciate that the attacker introduces a perturbation in the system. As a consequence, the Euclidean distance between the two motion devices starts to oscillate out of the expected behavior (w.r.t. Figure 4(a)).

When the attack is detected, the technique starts working and the reflective agent starts sanitizing the control commands to the moving agents to restore the nominal behavior of the system. Figure 4(e–f) show the winding graph of the motion devices under the approach. The attacked device corresponds to the vertically oriented ellipses. It is possible to observe some perturbations, due to the modifications introduced by the reflective agent when thwarting the adversary actions and recover the stability of the process. As a result, the spatial collision between the two devices is avoided and the system keeps working. Notice that the technique takes control of the physical environment in order to conduct the physical environment from an unstable behavior generated by the attack to a stable and safety behavior, converging to the normal behavior of the physical environment. Figure 4(c–d) show that approach neutralizes the effects of the attack right after a short period of instability. The approach does not eliminate the adversary. However, it contains the effects and reorients the system to the nominal case.

We argue that the solution is reflective since it creates a dynamic component at runtime to help with the function of the attacked control loops. In addition, the component is reflected to the network domain. It gets a greater control of the network than the victim component which has only the possibility to communicate through the network data plane. This is an advantage of the approach compared with other techniques such as redundancy, which implies to have a copy of the same component as a backup. In that case, the adversary may move the attack to the redundant component. For this same reason, routing-based mitigation techniques are not sufficient since the system may find an alternative route but the adversary may move to the new paths. Other solutions that implement mitigation at the node level, such as diversity, are not enough either since the adversary uses the network to perform the malicious activity. Cyber mechanisms to detect packet injection, such as, a Message Authentication Code (MAC), cannot mitigate this kind of attacks. Although they allow to drop modified messages, fail at satisfying real-time constraints. In our approach, the network itself is containing the adversary to revert its actions.

6 Conclusion

We have presented an attenuation approach driven by reflective programmable networking actions, in order to take control of adversarial attacks against Cyber-Physical Systems (CPS). The resulting CPS satisfies self-healing, i.e., during adversarial situations, it continues working in an autonomous way, while ensuring resilience. We have shown and validated the approach via experimental work. We have assumed a cooperation between two different families of controllers (e.g., feedback and reflective programmable networking controllers). Both cooperate together to reach a common goal (e.g, to ensure system stability). The two

controller families can have individual objectives (i.e., non-explicit objectives that are not conflicting between them, either). The obtained results are very promising. We plan to go beyond by expanding the analysis with additional controllers and adversaries. Further work will also include the achievement of optimal controller goals in terms of quality of control.

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, Nokia, BNP Paribas and the Regional Council of Brittany. The chair has been acknowledged by the Center of excellence in Cybersecurity. Authors acknowledge as well support from the European Commission (H2020 SPARTA project), under grant agreement 830892.

References

1. Albert, R., Jeong, H., Barabási, A.-L.: Error and attack tolerance of complex networks. Nature **406**(6794), 378 (2000)
2. Campbell, A.T., Katzela, I., Miki, K., Vicente, J.: Open signaling for ATM, internet and mobile networks (OPENSIG'98). SIGCOMM Comput. Commun. Rev. **29**(1), 97–108 (1999)
3. 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
4. Coulson, G., et al.: Reflective middleware-based programmable networking. In: The 2nd International Workshop on Reflective and Adaptive Middleware, pp. 115–119 (2003)
5. Elteto, T., Molnar, S.: On the distribution of round-trip delays in TCP/IP networks, pp. 172–181, November 1999
6. Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A.: Network configuration protocol (NETCONF) - internet engineering task force, RFC 6241, June 2011. http://www.ietf.org/rfc/rfc6241.txt
7. European Union Agency for Network and Information Security Agency (ENISA). Methodologies for the identification of Critical Information Infrastructure assets and services (2015). https://fullreportatwww.enisa.europa.eu/
8. Hachem, N., Debar, H., Garcia-Alfaro, J.: HADEGA: a novel MPLS-based mitigation solution to handle network attacks. In: 31st IEEE International Performance Computing and Communications Conference, IPCCC 2012, Austin, TX, USA, 1–3 December 2012, pp. 171–180 (2012)
9. Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Sean Wang, X.: Moving Target Defense: Creating Asymmetric Uncertainty For Cyber Threats, vol. 54. Springer, New York (2011)
10. Joolia, A., Coulson, G., Blair, G., Gomes, A.T., Lee, K., Ueyama, J.: Flexible programmable networking: a reflective, component-based approach (2003)
11. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
12. Krotofil, M., Larsen, J.: Rocking the pocket book: hacking chemical plants for competition and extortion. DEF CON, 23 (2015)

13. Kuipers, D., Fabro, M.: Control systems cyber security: defense in depth strategies. Technical report, Idaho National Laboratory (INL) (2006)
14. Lagu, S.S., Deshmukh, S.B.: Raspberry Pi for automation of water treatment plant. In: 2015 International Conference on Computing Communication Control and Automation (ICCUBEA), pp. 532–536, February 2015
15. Modbus Organization. Official Modbus Specifications (2016). http://www.modbus.org/specs.php. Accessed Apr 2019
16. Ogata, K., Yang, Y.: Modern Control Engineering, vol. 4. Prentice-Hall, Upper Saddle River (2002)
17. 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)
18. Queiroz, C., Mahmood, A., Tari, Z.: SCADAsim–a framework for building SCADA simulations. IEEE Trans. Smart Grid **2**(4), 589–597 (2011)
19. Rollins, M.: Beginning LEGO MINDSTORMS EV3. Apress, New York (2014)
20. 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
21. 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, August 2016
22. Rubio-Hernan, J., Rodolfo-Mejias, J., Garcia-Alfaro, J.: Security of cyber-physical systems. In: Cuppens-Boulahia, N., Lambrinoudakis, C., Cuppens, F., Katsikas, S. (eds.) CyberICPS 2016. LNCS, vol. 10166, pp. 3–18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61437-3_1
23. Rubio-Hernan, J., Sahay, R., De Cicco, L., Garcia-Alfaro, J.: Cyber-physical architecture assisted by programmable networking. Internet Technol. Lett. **1**, e44 (2018)
24. Sahay, R., Blanc, G., Zhang, Z., Debar, H.: Towards autonomic DDoS mitigation using software defined networking. In: SENT 2015: NDSS Workshop on Security of Emerging Networking Technologies, San Diego, CA, USA. Internet society, February 2015
25. Segovia, M., Cavalli, A.R., Cuppens, N., Garcia-Alfaro, J.: A study on mitigation techniques for SCADA-driven cyber-physical systems (position paper). In: Zincir-Heywood, N., Bonfante, G., Debbabi, M., Garcia-Alfaro, J. (eds.) FPS 2018. LNCS, vol. 11358, pp. 257–264. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18419-3_17
26. Soupionis, Y., Ntalampiras, S., Giannopoulos, G.: Faults and cyber attacks detection in critical infrastructures. In: Panayiotou, C.G.G., Ellinas, G., Kyriakides, E., Polycarpou, M.M.M. (eds.) CRITIS 2014. LNCS, vol. 8985, pp. 283–289. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31664-2_29
27. Teixeira, A., Shames, I., Sandberg, H., Johansson, K.H.: A secure control framework for resource-limited adversaries. Automatica **51**, 135–148 (2015)
28. Tennenhouse, D.L., Smith, J.M., Sincoskie, W.D., Wetherall, D.J., Minden, G.J.: A survey of active network research. Comm. Mag. **35**(1), 80–86 (1997)
29. The OMNeT++ network simulation framework. http://www.omnetpp.org/. Accessed Apr 2019
30. The OMNeT++/INET framework. http://inet.omnetpp.org/. Accessed Apr 2019
31. Varga, A., Hornig, R.: An overview of the OMNeT++ simulation environment. In: 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (Simutools) (2008)