2014 IEEE International Conference on Distributed Computing in Sensor Systems

Cyber-Physical Systems Attestation

Junia Valente, Carlos Barreto, Alvaro A. Cárdenas
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
Richardson, TX
Email: {juniavalente, carlos.barretosuarez, alvaro.cardenas}@utdallas.edu

I. Introduction

Cyber-Physical Systems (CPS) are monitored and controlled by a wide variety of sensors and controllers. The security of our cyber-physical critical infrastructures depends on the integrity of these devices and the software they execute; however, it has been repeatedly demonstrated that most of the devices interacting with the physical world (sensors and controllers) are extremely fragile to security incidents. The insecurity of these devices ranges from insecure-by-design implementations (e.g., devices that have a backdoor used for troubleshooting) to the inability to apply software updates to vulnerable devices.

One particular technology that can help us improve the trustworthiness of devices connected to the physical world is *attestation*. Attestation allows unauthorized changes to a device to be detected. Traditionally, attestation requires a hardware root-of-trust that can generate a certificate stating which software is currently running. Because most devices connected to the physical world do not have hardware-assisted roots of trust, and because most of them are embedded devices with processing power limitations, an alternative form of attestation without hardware-assisted measurements has been proposed in the form of *software attestation*. While software attestation can help a verifier check the integrity of devices, it still has several drawbacks that have limited their application in the field, like establishing an authenticated channel, the inability to provide continuous attestation, and the need to modify devices to implement the software attestation procedure.

In this paper we introduce a different form of attestation by exploiting the physical dynamics of the system. Our cyber-physical attestation proposal has several advantages over software attestation including the ability to be deployed without modifying legacy devices (or devices that would need to be re-certified if they are modified), the potential to perform attestation continuously, and the fact that you do not need to establish authenticated channels to the devices (only to the physical world). While CPS-attestation promises many advantages it has a couple of drawbacks as well; most notably, the fact that it provides a *weak attestation* in the sense that the verifier cannot prove that a device has not been compromised, only that it continues to work according to its expected behavior.

II. Attestation

A. Remote Attestation

Remote attestation is a trust establishment mechanism that allows a platform (the *attestor*) to reliably voucher on its current state to a remote verifier (the *challenger*). This mechanism allows the challenger to obtain confidence in the trustworthiness of the attestor by means of hardware-based security mechanisms. Typically, remote attestation assumes that the attestor is a trusted platform which contains a Trusted Platform Module (TPM). Thus, it is capable of collecting integrity measurements during boot time and securely storing these measurements in the TPM's Platform Configuration Registers (PCRs). During attestation, the challenger requests the attestor to send the current configuration measurements. In response, the attestor uses the TPM to sign the configuration measurements and sends the measurements to the challenger. The challenger verifies the signature and compares the received platform configuration measurements against expected measurements [1].

A major limitation of this approach is that it is difficult for the challenger to comprehensively know every possible platform configuration measurements to verify the attestor. Another drawback is that it may reveal too much information about the platform configuration thus making attacks on the platform easier. Also, this approach is not feasible to be used in resource constrained devices that cannot afford to have extra dedicated hardware. For more information on bootstrapping trust in general purpose computers refer to [2].

B. Software Attestation

Software attestation targets resource-constrained devices that cannot afford to implement security defenses assisted by dedicated hardware such as the TPM. Software attestation is different from remote attestation which relies on trusted computing principles (i.e., *hardware roots-of-trust*, *trusted boot*, *protected storage*) and shared secrets between the prover and verifier.

Software attestation is a challenge-response technique that enables a *verifier* to check the integrity of the memory contents of another device (the *prover*) against modifications by malicious code. In this approach, the verifier relies on the *computation time* the prover takes to compute the response to decide whether the prover has been compromised [3]. This approach has been used to verify code integrity of sensor

978-1-4799-4618-1/14 $31.00 © 2014 IEEE
DOI 10.1109/DCOSS.2014.61

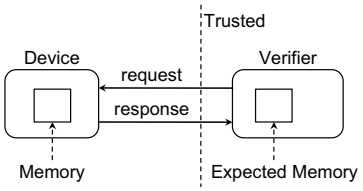

Fig. 1: Traditional Software Attestation

network devices [4], Remote Terminal Units [5], and firmware on peripherals [6].

Fig. 1 shows a generic attestation protocol between a verifier and a prover, denoted by V and P respectively. The main goal of V is to check whether the memory content of P is as expected. We assume that the verifier is a trusted system and has knowledge of the expected memory content of P. During the attestation process, V sends a randomly generated challenge and requests the device to *attest* on its current state $State(P)$. The *verification procedure* succeeds only if the memory content is the same as some expected response (i.e., $State(P) = S$), and fails with high probability if the content differs even by a single byte [4]. We assume that the prover contains a verification procedure prior to the attestation.

While software attestation can be applied to embedded systems, recent research [7] suggests shortcomings to existing software attestation approaches as it may be possible for malicious code to hide itself from an attestation through *return-oriented programming* or *compression attacks*. In addition, traditional attestation mechanisms can be vulnerable to *time-of-check-to-time-of-use* (TOCTOU) attacks; i.e., it is possible for an attacker to leverage the elapsed time between the verification procedure and the use of the device. Furthermore, software attestation relies on an optimal implementation of the verification algorithm: it should be hard to find another implementation of the algorithm such that the prover can execute in significantly less time. Otherwise, an attacker could use a faster implementation and take advantage of the time difference to perform extra computations (e.g., lie about its state) [3]. However, it is difficult to design such procedure based on tight timing constraints.

III. Cyber-Physical Systems Attestation

A. Overview

In this paper we propose *CPS-attestation* as an attestation technique for control systems to attest their state to an external verifier. During CPS-attestation, the verifier introduces *false* control signals to the system and observes the system dynamics to verify that the sensors and controllers are operating correctly. We assume that the verifier is a trusted system and has a correct mathematical model of how the control system *should* behave.

The architecture of a process control monitored through a Process Control System (PCS) or Supervisory Control and Data Acquisition (SCADA) system is shown in Fig. 2. In this case, the programmable logic controller (PLC) implements a control law that manages the plant behavior. The control actions are calculated based on the sensor measurements

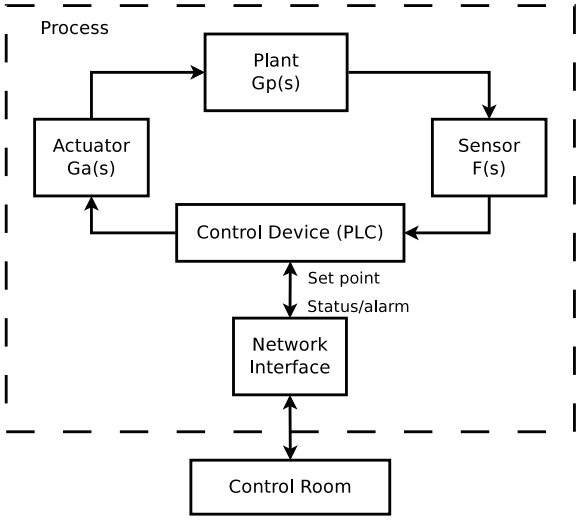

Fig. 2: Structure of a Control System

and the desired system state (set point). Furthermore, the control actions are carried out though an actuator, that interacts directly with the plant. The desired system state is a parameter fixed remotely by a central agent that might be a supervisory control system or an operator.

CPS-attestation enables a verifier to continuously monitor the dynamics of the control system over time and detect whether a component is not behaving as expected or if it is driving the system to an unsafe state (See Fig. 3). The main purpose of this approach is to design the CPS-attestation protocol in such a way that we maximize the probability for detecting an attack (if an attack indeed exists) while minimizing the side effects on the physical system.

CPS-attestation inherits characteristics from software attestation because it does not rely on dedicated hardware nor on cryptography techniques. It is however different from software attestation in that CPS-attestation does not rely on a verifiable proof of the component trustworthiness to detect a compromise: the integrity of individual components (e.g., sensors, actuators, controllers) is implicitly verified over time. This approach gives us an intuition of the correctness of the code running in these components by verifying whether the entire system behaves as expected.

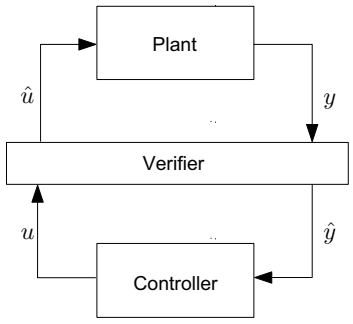

Fig. 3: CPS-Attestation Conceptual Model

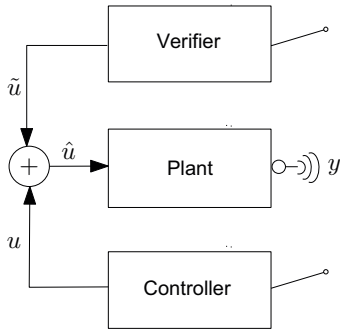

Fig. 4: Practical CPS-Attestation Implementation

While having a verifier that mediates all access to the physical system and can act as a man-in-the-middle for all communications between the physical system and the controller (e.g., a PLC) is a good conceptual model; implementing this ideal specification would be very challenging in practice.

First, existing communications between sensors and controllers would need to be intercepted. The communications would no longer be end-to-end, and the verifier would need to act as a gateway between communications.

Similarly, communications between the controller and the actuator (which most of the times are through an analog signal) would again need to be intercepted and modified.

Therefore we argue that a practical implementation of CPS-attestation can be done by listening to the same wireless sensor signal as the controller, and at the same time, tapping into the analog actuation channel to inject a control perturbation, as shown in Fig. 4. We emphasize that the verifier will need to have a physical model of the plant and a model of the control algorithm to be able to identify false sensor or controller signals. The perturbation can detect a false sensor measurement by driving the system to a state where the sensor will not respond appropriately. If a sensor signal is verified to work correctly, then a compromised controller sending false control signals can be detected by injecting a perturbation signal and then verifying that the system will be controlled properly by the PLC control signal.

B. CPS-attestation advantages over traditional attestation mechanisms

Purpose. The main purpose of CPS-attestation is to verify the integrity of a control loop, i.e., detect if a controller or a sensor is not behaving as expected or if it is driving the system to an unsafe state. This is different from traditional attestation techniques that rely on verifiable proofs such as secure measurements stored in Platform Configuration Measurements (PCRs) of a TPM during a trusted boot, or current memory contents in resource constrained devices used by software attestation.

No Prover Modification. Similar to other attestation mechanisms, CPS-attestation relies on two main components: the verifier and the prover. In traditional attestation we necessarily need to deploy new provers with dedicated hardware (e.g., TPM) or modify the prover to have the software-based attestation code (the verification procedure). An important difference and major advantage of CPS-attestation is that the provers (system components like PLCs and sensors) do not require hardware nor software modification and therefore CPS-attestation can be applied to systems that cannot be patched (e.g., because patching would violate their certification), to legacy devices, or to systems that do not need to be replaced by new hardware (see Fig. 5).

	Hardware modification	**Software modification**
Remote attestation	yes	yes
Software attestation	no	yes
CPS-attestation	**no**	**no**

Fig. 5: Modification Requirements to Attesting System

Continuous Attestation. CPS-attestation is a promising solution to verify control systems against attacks because it leverages the nature of control systems to enable an attestation that is *continuous*. In traditional approaches, attestation is the verification of a device at a given point in time; however, an attacker can compromise a device after this verification. Furthermore, software attestation does not necessarily say anything about how trustworthy a device has been in the past. In short, traditional attestation approaches can only verify trustworthiness of a device at particular points in time, which poses significant limitations, in particular when it may be predictable to know when the attestation protocol will occur. CPS-attestation can (if desired by the verifier) verify the system continuously.

No Authenticated Channel to Provers. Hardware-assisted attestation allows for *remote* attestation because an authenticated channel can be established by verifying that the device the verifier is connecting to knows the secret keys stored in their TPM. For software attestation creating authenticated channels is more difficult and it might require a visual authentication and a short-range communication protocol to prevent communication latencies. CPS-attestation does not need an authenticated channel to the provers. It only needs an authentic channel to the physical system (by e.g., tapping to the analog actuation signal), and (because most modern sensor networks such as those supported by WirelessHART, ISA100 or IEEE 802.15.4g are wireless) it just needs to listen passively to the same wireless transmission that the controller is listening to.

Implicit Attestation. Software attestation requires the device to dedicate all its resources during the attestation protocol. This can be problematic to real-time systems that require sensor and controllers to be responsive to system conditions at all times. CPS-attestation is done implicitly by monitoring the behavior of the system to disturbance signals; and thus, it does not need provers to stop their normal operation to perform the attestation.

C. CPS-attestation limitations

Weak Trust. As long as the system works as expected (by reliably comparing the outcome of the system) it is not possible to detect the presence of a compromise. This is different from more strict approaches that aim to build trustworthy systems that can reliably attest on their integrity (e.g., truthfully report whether the system is in a good state or in a bad state).

Additional Devices. While CPS-attestation does not require any modification of the provers, it still requires additional trusted devices in the control loop to verify the system under control.

Process Disturbance. The verifier will act as a process disturbance by injecting an external control signal to drive the system towards a desired state and then monitor the response by the sensor and the controller. This process disturbance will result in higher operational costs (more energy to change actuator status and will also drive the system further away from the desired setpoints). As such, a research challenge is to design a disturbance signal $\hat{u}$ such that we maximize the probability of detecting an attack (in the case a compromise exists) and minimize the side-effects on the process.

IV. Related Work and Research Challenges

Our goal in this position paper was to initiate the discussion on the suitability of CPS-attestation and the associated research challenges. CPS-attestation is similar to previous work on fault-detection [8], attack detection [9], resilient control [10] and resilient estimataion [11]. In particular previous work on detecting replay attacks [12] is similar to our proposal in the sense that the authors use a noisy control signal to detect sensor replay attacks. The main drawback of their study is that the security analysis is missing: i.e., the main difference between fault detection, and attack-detection is that in an attack scenario we need to assume that an attacker knows the details of our intrusion detection mechanism and will strategically select its attack signals in order to bypass our detection methodology.

CPS attestation is an extension to previous related work on control theory because it brings the traditional security framework of attestation in order to pose the problem of verification of devices by actively sending false control signals. The main open research challenge will be in the characterization of its security (which attacks cannot be detected and why) and its performance impact (we do not want the verification mechanism to impose significant overhead to the operation of a control system).

References

[1] A. Lee-Thorp, "Attestation in trusted computing: Challenges and potential solutions," *Royal Holloway University of London*, vol. 31, 2010.

[2] B. Parno, J. M. McCune, and A. Perrig, *Bootstrapping Trust in Modern Computers*, ser. Springer Briefs in Computer Science. Springer, 2011, vol. 10.

[3] F. Armknecht, A.-R. Sadeghi, S. Schulz, and C. Wachsmann, "A security framework for the analysis and design of software attestation," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 1–12.

[4] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: softWare-based attestation for embedded devices," in *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, May 2004, pp. 272–282.

[5] A. Shah, A. Perrig, and B. Sinopoli, "Mechanisms to provide integrity in SCADA and PCS devices," in *Proceedings of the International Workshop on Cyber-Physical Systems-Challenges and Applications (CPS-CA)*, 2008.

[6] Y. Li, J. M. McCune, and A. Perrig, "VIPER: verifying the integrity of PERipherals' firmware," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 3–16.

[7] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, "On the difficulty of software-based attestation of embedded devices," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 400–409.

[8] P. Menon and C. Edwards, "A sliding mode fault detection scheme for corrupted measurement data exchange in a network of dynamical systems," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Dec 2013, pp. 2852–2857.

[9] F. Pasqualetti, F. Dorfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *Automatic Control, IEEE Transactions on*, vol. 58, no. 11, pp. 2715–2729, Nov 2013.

[10] C. Kwon and I. Hwang, "Hybrid robust controller design: Cyber attack attenuation for cyber-physical systems," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Dec 2013, pp. 188–193.

[11] Y. Mo and B. Sinopoli, "Robust estimation in the presence of integrity attacks," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Dec 2013, pp. 6085–6090.

[12] Y. Mo, R. Chabukswar, and B. Sinopoli, "Detecting integrity attacks on scada systems," pp. 1–1, 2013.