Poster: Re-thinking the Honeypot for Cyber-Physical Systems

Samuel Litchfield*, David Formby*, Jonathan Rogers†, Sakis Meliopoulos*, Raheem Beyah*
*School of Electrical and Computer Engineering
Georgia Institute of Technology
Email: {slitchfield3, djformby, sakis.m, rbeyah}@gatech.edu
†School of Mechanical Engineering
Georgia Institute of Technology
Email: jonathan.rogers@me.gatech.edu

I. Introduction

Since the early 1990s, the idea of entrapping and deceiving computer attackers in order to study their behavior and misdirect them has been used with great success in the computer security field. This practice, traditionally done through a computing system configured to emulate critical resources, called a Honeypot, has revealed attacker strategies and kept more critical computing resources safe.

As the Cyber-Physical Systems (CPS) space grows and becomes increasingly networked, attackers have been more interested in compromising the resources controlling these systems. In response, honeypots have been designed to emulate CPS specific components. However, *all* existing CPS honeypots neglect certain aspects of these systems that can alert an attacker to the nature of the honeypot, namely the simulation of the attached *physical process* and the *physics of the devices* that interact with the process. We propose a new CPS specific Honeypot framework, called HoneyPhy: A Physics-aware Honeypot Framework, that addresses these problems and aims to be extensible to all CPS.

II. Why Existing CPS Honeypots Are Not Sufficient

In traditional network focused honeypots, as well as in existing CPS honeypots [1]–[5], the main goal was to emulate the kinds of protocol quirks that fingerprinting utilities like Nmap and p0f look for. However, CPS honeypots should provide auxiliary information arising from the attached physical system. This auxiliary information is both the ability to compare the moment to moment state of the CPS for consistency (i.e., leveraging the physics of the process and sensors), as well as observing the individual connected devices for unreasonable actuation times. If either the process physics or device actuation time are unrealistic, an attacker can easily determine if they are in a honeypot.

This is illustrated in Figure 1, showing an attacker interacting with a simple HVAC system at varying levels of modeling. The left scenario illustrates an attacker interacting with a real system, so all delays and responses result from process and device behavior. The right scenario illustrates an attacker interacting with a CPS honeypot that does not attempt to model process behaviors and device delay, and the lack of this delay and deviations from expected process behavior will alert the attacker to the honeypot. The middle scenario illustrates an attacker interacting with a CPS honeypot that is capable of modeling *both* process behavior and device delay. Responses provided to the attacker are thus realistic, and the attacker proceeds to perform observed malicious actions.

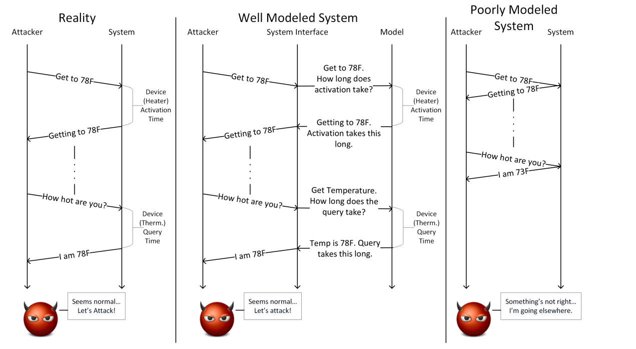
Reality
Attacker
System
Get to 78F
Device (Heater) Activation Time
Getting to 78F
How hot are you?
Device (Therm.) Query Time
I am 78F
Seems normal... Let's Attack!
Well Modeled System
Attacker
System Interface
Model
Get to 78F
Get to 78F. How long does activation take?
Device (Heater) Activation Time
Getting to 78F. Activation takes this long.
Getting to 78F
How hot are you?
Get Temperature. How long does the query take?
Device (Therm.) Query Time
Temp is 78F. Query takes this long.
I am 78F
Seems normal... Let's attack!
Poorly Modeled System
Attacker
System
Get to 78F
Getting to 78F
How hot are you?
I am 73F
Something's not right... I'm going elsewhere.

Fig. 1. Outcomes of an attacker interacting with different levels of modeling

III. What Level of Interaction Should a CPS Honeypot Exhibit?

Pure high-interaction honeypots are fundamentally unsuited to CPS, because they rely on either deploying another physical copy of the resource in question, or somehow virtualizing it. Deploying a copy of an entire CPS with the purpose of being compromised exposes the same safety risks as the original system, and imposes large costs. A pure high-interaction honeypot can be deployed for a single component within a CPS, but without the physical portion of the system to interact with, the usefulness of these honeypots is limited. One solution, proposed in more detail below, is to create a *hybrid-interaction honeypot*, where real CPS devices and interfaces interact with process and device simulations that can effectively replicate the behavior of the CPS process.

IV. Our Vision for Future CPS Honeypots

Our vision for future CPS honeypots addresses the limitations of current CPS honeypots by providing an extensible framework, HoneyPhy: A Physics-aware honeypot framework, for accounting for the physics of the physical process and the mechanical delays of the physical actuators.

Future honeypots should, as existing honeypots already do, correctly model software and protocol fingerprints. This interface layer of the honeypot could be either high or low interaction, depending on access to equipment such as Human-Machine Interfaces (HMIs).

In addition to this, future honeypots should correctly model the behavior of the physical system. This primarily ensures that physical parameters, when queried, behave in a way consistent with attacker expectations.

Finally, future honeypots should correctly model the operating time delays introduced by the constituent devices within the CPS. These operation times can be modeled, and these models can be generated in one of two ways, white box modeling or black box modeling [6].

V. Proposed New CPS Honeypot Architecture

To satisfy this vision, HoneyPhy, a new CPS hybrid-interaction honeypot framework is proposed. The new framework is composed of three major components: the Internet Interface Module, the Process Model Module, and the Device Model Module. Each module's contents, permissions, and metadata are configured by a central XML file. A framework overview can be seen in Figure 2.

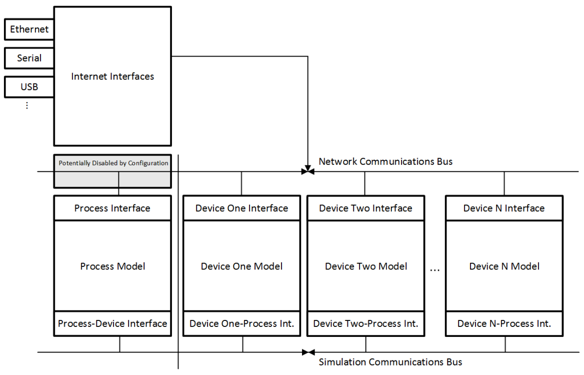

Fig. 2. Proposed CPS Honeypot Architecture

A. Internet Interface(s) Module

The Internet Interface Module exposes the declared interfaces at the declared addresses. It will maintain connections and multiplex them to their destinations while ensuring outgoing packets reflect the network fingerprint for each device and modifying them if necessary.

B. Process Model(s) Module

The Process Model exists to simulate the physical process in question. It can be interrogated and acted upon by the other devices modeling sensors and actuators, and should simulate the process in real time.

The Process Model communicates with the various Devices and Models over a separate databus. If desired, a secure Internet-facing interface can be opened to remotely interact with the process model directly.

Process Models could consist of LabView simulations, replays of empirically observed responses, or more traditional controls system models such as a Linear Dynamical System or Auto-Regressive models.

C. Device Model(s) Module

The Device Models encompass all devices found within a CPS. Where they model computing devices such as PLCs, the model should implement logic to simulate those devices. This can include interpreting incoming queries and responding with the value they obtain by querying the process model, or executing incoming commands by modifying the process model. Where Device Models simulate mechanical devices such as relays or valves, the model should introduce a suitable delay corresponding to the time necessary to change the devices state.

Device models can range from very simple low level black box timing models to real devices, sufficiently instrumented to interact with the process model.

D. Inter-module Communication

While our framework does not specify a required inter-module communication method, it does specify where communication must be available. The Internet Interface module must be able to route incoming/outgoing communications to all applicable devices, and optionally expose the internet facing interface for the process model. Additionally, all devices must be able to talk not only to each other, but also to the process model. This necessitates a separate Process/Device databus.

VI. Conclusion

This poster presents an easily extensible framework for creating CPS honeypots, called HoneyPhy. This was motivated by the lack of, or difficulty in adding, process and device simulation in extant CPS honeypots, and the resulting information leakage. HoneyPhy aims to make these simulations as easy to implement as possible, in an effort to spread the adoption of convincing CPS honeypots.

More details and future updates to HoneyPhy can be found at honeyphy.gatech.edu.

References

[1] V. Pothamsetty and M. Franz. (2005) Scada honeynet project: Building honeypots for industrial networks. [Online]. Available: http://scadahoneynet.sourceforge.net/

[2] Digital Bond. (2016) Scada honeynet. [Online]. Available: http://digitalbond.com/tools/scada-honeynet/

[3] K. Wilhoit and S. Hilt, "The gaspot experiment: Unexamined perils in using gas-tank-monitoring systems," 2015. [Online]. Available: https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/whitepapers/wp_the_gaspot_experiment.pdf

[4] L. Rist, J. Vestergaard, D. Haslinger, and A. Pasquale. (2016) Conpot. [Online]. Available: http://conpot.org

[5] D. Buza, F. Juhasz, M. Felegyhazi, and T. Holczer, "Cryplh: Protecting smart energy systems from targeted attacks with a plc honeypot," in *Smart Grid Security: Second International Workshop*, 2014, pp. 181–192.

[6] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems," *NDSS*, Feb 2016.