Software-Defined Cyber-Physical Systems to Support Mobile-Defense Security of Critical Utilities

Position Paper submitted to 2013 NSF Workshop on Energy Cyber-Physical Systems

Riccardo Bettati, Guofei Gu, Narasimha Reddy
Texas A&M University

Background

A number of highly visible recent attacks on cyber-physical systems, such as SCADA and other control and automation infrastructures (most prominently the Stuxnet worm,) make implicit use of the *location* of the victim in the attacked system. This form of *location sensitivity* is prevalent in SCADA-like systems, as proximity of the supervisory component to the controlled or monitored device is necessary. For example, Stuxnet exploited vulnerabilities in the Siemens WinCC operator control and monitoring system to then exert control on the S7-300 programmable logic controller (PLC), which in turn allows control of the frequency converters of the plant. Existing known vulnerabilities in other SCADA systems, such as C3-ilex's EOScada and a variety of PLC's, allow for similar scenarios.

Because of this proximity requirement, traditional security mechanisms have relied to a great extent on isolation and on physical security. The recent exploits illustrate that physical security is not nearly sufficient: The LNK exploit to distribute Stuxnet to disconnected control hosts uses flash drives as attack vector. Similarly, there is growing evidence that service and maintenance personnel are connecting infected laptops to SCADA and other automation systems as part of servicing and monitoring procedures.

At the same time, the systems targeted by this type of attacks are highly complex and multilayered. They are composed of a variety of software from different vendors, which interact and so offer a particularly rich and vulnerable attack surface, easily overwhelming the system operators[1].

Location Metamorphism

We propose *Location Metamorphism* to address both the location sensitivity of attacks and the demand for manageability of complex systems by overtaxed personnel. By *Location Metamorphism* we mean that the *location* of a system component (typically a server or client program, a dynamically linked library, or similar) can vary over the lifetime of a system. Throughout the system life cycle, it can be hosted by a (typically unpredictable) sequence of hosts. Through appropriate routing of requests, the component(s) of the system can migrate in response to external influences (such as, triggered by disasters or attacks,) or controlled by policy decisions, such as changes in defensive posture, or proactively, in order to perturb the attack surface.

For example, in a location-metamorphic system one could co-locate and aggregate critical components of a utilities infrastructure into virtual private clouds and so benefit of economy of scale to protect the system.

Challenges

In order to make this type of location metamorphism beneficial, a number of requirements must be met:

First, transitioning applications from a local host to remote servers must lead to a net reduction of the attack surface, both in the local host, the remote server, and the enabling network infrastructure. This includes (a) attack surface provided by the application itself, (b) vulnerabilities during the transition, and (c) any vulnerabilities provided by the location metamorphism support system.

Similarly, migration and redirection mechanisms must support hybrid systems, which include both "cyber" and "physical" components. While software stacks and protocols can be easily shimmed to support location metamorphism of cyber components, this is less the case for physical components. Protocols need to be investigated that are cognizant of the cyber-physical aspects of the system. In particular, they must maintain latency tolerances expected by users and required by real-time components (such as monitoring devices and generally controllers).

Further, basic operations to support location metamorphism must be provided. Appropriate mechanisms must be in place to *transition* (that is, effectively *migrate*) applications or application components.

[1] According to the U.S. Census Bureau, in 2008, 70% of all utilities companies in the U.S. employed less than 20 employees, with an additional 23% employing less than 100 employees. It is unlikely, therefore, that sufficient resources can be devoted to cyber protection of the utilities infrastructure.

Finally, the manageability of location-metamorphic applications must be ensured.

Software-Defined Cyber-Physical Systems

Underlying the challenges to support location-metamorphic cyber-physical systems is the need to de-couple the application (and the communication supporting it) from the underlying physical system, comprising network, computational and security infrastructure, and sensor and actuation devices.

In this position paper we propose to extend the "Software-Defined X" paradigm (software-defined networking, software-defined data center) from a traditional enterprise environment to systems that comprise of traditional computing and network components as well as physical components.

Architecturally, we envision side-by-side frameworks to address the control planes of (a) the network, (b) the application-components, and (c) the access and QoS management of physical components.

A number of technical hurdles that need to be addressed in such a setting are well known from traditional real-time and cyber-physical systems: (a) How to provide end-to-end hard-real time constraints during normal operations and during large-scale mode changes that would be triggered by component migrations and other system reconfigurations. (b) How to represent, map, and satisfy QoS requirements (timing, connectivity, and availability) that are incompatible with large-scale deployments because they have been specified with co-location in mind.

Novel challenges include: (a) How to secure the frameworks that implement the software realizations of the control planes and their interactions. (b) How to triage access to physical components as a result of overload, malfunction, or attacks. (c) How to generally "proxify" non-virtualizable but timing-sensitive components.