Sustainable Agile Security Enabled by Systems Engineering Architecture

Rick Dove, rick.dove@incose.org

How can we field systems with long life expectancies, embedding security that can deal with attacks that won't be invented until a year or more after systems are put into service? This article offers a path based on an agile architecture. But first, a quick review of a recent incident will set some context.

The now infamous Stuxnet attack (Kushner 2013) ushered in a new era of attack sophistication, with four vulnerabilities exploited that had not been used before, so-called zero-day attacks. The victim system was Iran's Natanz nuclear-fuel enrichment plant, with thousands of COTS (commercial off-the-shelf) centrifuges and many SCADA (supervisor control and data acquisition) COTS computers that controlled them. The complete story may never be known, but a more than sufficient portion of the attack method has been reverse engineered, and is now in the public domain—for use by others less resourceful.

There was an "air gap" between the Natanz plant systems and the outside world—it was an isolated internal network with no external network connections. There was apparent confidence that the air gap would prevent an intrusion. But insider folly or insider maliciousness appears to have introduced an infected USB device from the outside. Then, too, there were SCADA computers purchased off the shelf, and placed in a network of communicating devices that likely trusted each other. Even if the SCADA equipment was deeply examined upon arrival, an infection occurring after installation could spread among naïvely trusting networked equipment, unquestioned.

Note that Stuxnet successfully attacked a cyber-physical system, targeting the centrifuges, not the information system. Once implanted, Stuxnet laid dormant most of the time, coming to life once a month for a short period of time, instructing the centrifuges to spin at rates outside their specifications, damaging them over a period of time. While active, Stuxnet spoofed the monitoring system just like the television spy shows, showing normal operating data to the monitors that didn't reflect what was actually occurring. This slow, stealthy, periodic attack over many months just looked like accelerated wear, and didn't cause an immediate investigation alert.

How can we mitigate the inevitable never-seen-before threats to fielded systems? Here's how.

System Architecture is Where Security Enablement Starts … or Stops

Architecture focuses on the high-level allocation of responsibilities between different components of the system, and defines the interactions and connectivity between those components. Responsibility for security requirements established during the requirements processes should be allocated to functional components and security-specialty components as appropriate during the architectural design process.

System resilience permits a system to operate while under attack, and to recover afterwards. The system may end up with degraded performance, but it will continue to deliver its critical functionality. Resilience is an architectural feature that is difficult to provide later in the development lifecycle, and very costly after deployment.

Long-life systems will have functional upgrades and component replacements throughout their life. Insider threats and supply-chain threats may manifest as components with embedded malicious capabilities that may lie dormant until activated on demand. This argues for self-protective system components that distrust communications and behaviors of interconnected components, rather than relying on system-perimeter protection or trusted environment expectations.

The Concept of Operations

Continuous evolution of system security is necessary to maintain parity with a continuously evolving threat environment. This requires the ability to respond effectively under unpredictable and uncertain circumstances, as often as necessary.

Enabling continuous system security evolution requires an agile-system concept of operations, one that recognizes the need for effective asynchronous system-security change. A fundamental agile architecture pattern enables this, and will be recognized in a simple sense as a modular drag-and-drop, plug-and-play architecture, with some critical aspects not often called to mind with the general thoughts of a modular architecture.

Need—A cross-industry study for the US Office of Naval Research in 1991 (Nagel 1992) observed that technology and the environment in which it is deployed were co-evolving at an increasing rate, outpacing the adaptation capabilities of most organized human endeavors. Agility, as a systemic characteristic, was identified as a new need generally missing in systems we call enterprise, the systems supporting enterprise, and the systems produced by enterprise. Decreasing relevance and life expectancy for traditional systems of all kinds create the counterbalancing need for agility.

Definition—Agility is the ability of a system to thrive in an unpredictably evolving environment; deploying *effective response* to both opportunity and threat, within mission.

Metrics – Effective response has four metrics: timely (fast enough to deliver value), affordable (at a cost that can be repeated as often as necessary), predictable (can be counted on to meet the need), and comprehensive (anything and everything within the mission boundary)..

Value Proposition—Risk management in an evolving unpredictable environment is the value proposition for agile systems. An agile system is constructed to enable and facilitate augmentation, reconfiguration and scalability of reusable assets in response to unpredictable situations, and agility is sustained with active management of responsibilities that constantly evolve the agility-enabling capabilities.

ConOps—In short, the system ConOps should call out the ability to reconfigure and augment system security throughout the development and operational lifecycle of the system, and it should call out the need for rapid reconfiguration of security at the system level (or both the system-of-systems level and the system level in a system of systems). Figure 1 depicts a notional architectural concept.

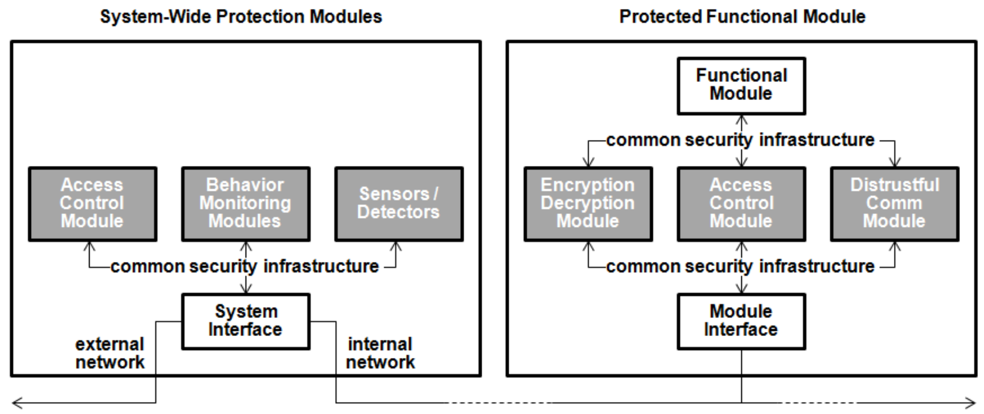
System-Wide Protection Modules
Protected Functional Module
Access Control Module
Behavior Monitoring Modules
Sensors / Detectors
common security infrastructure
System Interface
external network
internal network
Functional Module
common security infrastructure
Encryption Decryption Module
Access Control Module
Distrustful Comm Module
common security infrastructure
Module Interface

Figure 1. Notional concept – common security infrastructure enables rapid evolution; other security modules are likely include, e,g,. functional-module behavior monitoring, etc.

Fundamentals

This section focuses on fundamental needs, definitions, and necessary and sufficient enabling concepts for agile systems of any kind (Dove 2001, Dove 2002)—most especially agile security functionality.

Architecture—Achieving good agile response metrics is enabled or hindered by architecture. One fundamental Agile Architecture Pattern has emerged from extensive investigation and appears both necessary and sufficient. It will be recognized in a simple sense as a loosely coupled, drag-and-drop, plug-and-play architecture, with some critical aspects not generally called to mind with the general thoughts of a modular architecture. There are three critical elements in the architecture: a catalog of drag-and-drop *encapsulated* modules and the module pools in which they belong, a catalog of the passive infrastructure rules and standards that enable and constrain plug-and-play operation, and the designation of an active infrastructure of four specific responsibilities that sustain agile operation.

- Modules—Modules are self contained encapsulated units complete with well-defined interfaces which conform to the plug-and-play passive infrastructure. They can be dragged-and-dropped into a configuration of response capability, with relationship to other modules determined by the passive infrastructure. Modules are encapsulated so that their methods of functionality are not dependent on the functional methods of other modules, except as the passive infrastructure dictates.
- Passive Infrastructure—The passive infrastructure provides drag-and-drop connectivity between modules. Its value is in isolating the encapsulated modules so that unexpected side effects are minimized and new operational functionality is rapid. Selecting passive infrastructure elements is a critical balance between requisite variety and parsimony—just enough in standards and rules to facilitate module connectivity, but not so much to overly constrain innovative system configurations. Passive infrastructure typically evolves, but slowly; generally when migration to next generation capability is appropriate.
- Active Infrastructure—An agile system cannot be designed and deployed in a fixed event and then left alone. Agility is most active as new system configurations are assembled in response to new requirements – something which may happen very frequently, even daily in some cases. Four responsibilities are required and must be designated and embedded within the system to ensure that effective response capability is possible at unpredictable times.
    - Module Mix/Evolution—Who (or what process) is responsible for ensuring that new modules are added to the roster, and existing modules are upgraded, in time to satisfy response needs?
    - Module Readiness— Who (or what process) is responsible for ensuring that sufficient modules are ready for deployment at unpredictable times.
    - System Assembly— Who (or what process) assembles new configurations when new situations require something different in capability?
    - Infrastructure Evolution—Who (or what process) is responsible for evolving the passive and active infrastructures as new rules and standards are anticipated and become appropriate.

Figure 2 depicts a graphical representation of the architecture.

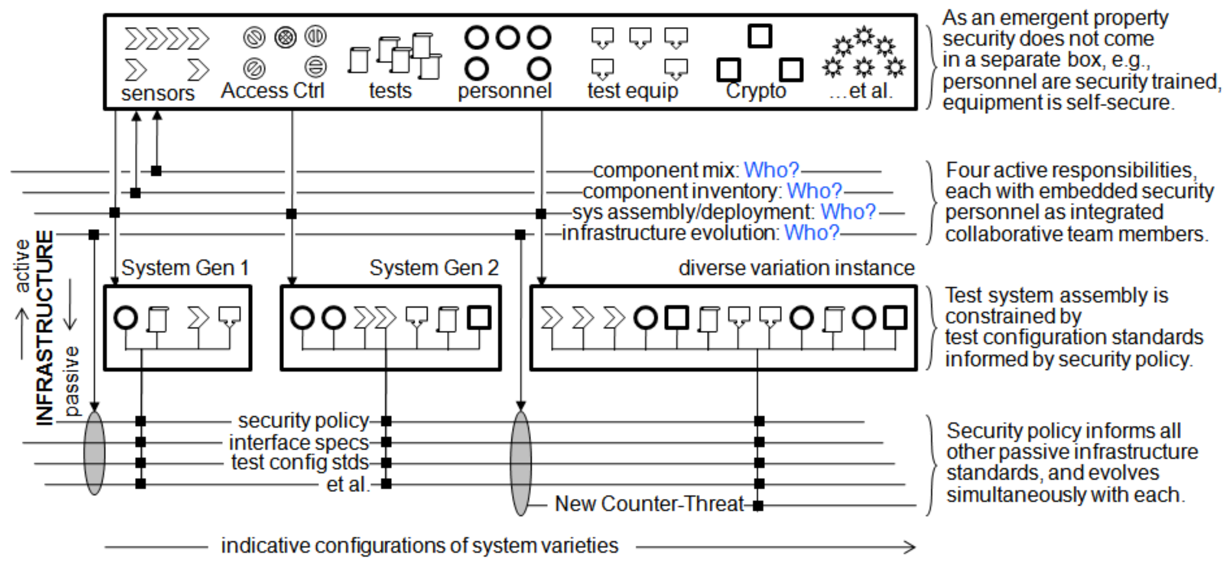
sensors
Access Ctrl
tests
personnel
test equip
Crypto
…et al.
As an emergent property security does not come in a separate box, e.g., personnel are security trained, equipment is self-secure.
component mix: Who?
component inventory: Who?
sys assembly/deployment: Who?
infrastructure evolution: Who?
Four active responsibilities, each with embedded security personnel as integrated collaborative team members.
active
INFRASTRUCTURE
passive
System Gen 1
System Gen 2
diverse variation instance
Test system assembly is constrained by test configuration standards informed by security policy.
security policy
interface specs
test config stds
et al.
New Counter-Threat
Security policy informs all other passive infrastructure standards, and evolves simultaneously with each.
indicative configurations of system varieties

Figure 2. Architecture for Security Reconfiguration/Augmentation/Evolution

Principles—Ten Reusable-Reconfigurable-Scalable design principles add the flesh to the bones of the architecture, and are briefly itemized here.

Reusable Principles:

- Encapsulated Modules— Modules share well defined interaction and interface standards, and are easily inserted or removed in system configurations.
- Facilitated Interfacing (Plug Compatibility)—Modules are reusable and replicable, with supporting facilitation for finding and employing appropriate modules.
- Facilitated Reuse—Modules are reusable and replicable, with supporting facilitation for finding and employing appropriate modules.

Reconfigurable Principles:

- Peer-Peer Interaction—Modules communicate directly on a peer-to-peer relationship, and parallel relationships are favored, rather than sequential relationships are favored ones.
- Distributed Control and Information – Modules are directed by objective rather than method, decisions are made at point of maximum knowledge, and information is associated locally but accessible globally.
- Deferred Commitment—Requirements can change rapidly and continue to evolve. Work activity, response assembly, and response deployment is deferred to the last responsible moment, to avoid costly wasted effort that may also preclude a subsequent effective response.
- Self-Organization—Module relationships are self-determined where possible, and module interaction is self-adjusting or self-negotiated.

Scalable Principles

- Evolving Standards—Passive infrastructure standardizes inter-module communication and interaction, defines module compatibility, and is evolved by designated responsibility for maintaining current and emerging relevance.
- Redundancy and Diversity—Duplicate or replicable modules provide capacity right-sizing options and fail-soft tolerance, and diversity among similar modules employing different methods is exploitable.

- Elastic Capacity—Modules may be combined in responsive configurations to increase or decreased functional capacity within the current architecture.

Requirements—In addition to the system functional requirements, response situation analysis helps identify response requirements that inform the design of architecture, indicating the necessary nature of modules and module pools, which in turn help identify the necessary nature of both passive and active infrastructure. Requirements arising from response situational analysis may not be directly present in customer requirements, but are necessary for effective architecture design. Unlike functional requirements typically captured in all-encompassing specific "shall" statements, response requirements need only enumerate sufficient diversity to result in a capability that can respond to un-enumerated situations. An effective framework for structuring response situational analysis drives analytical thinking in four reactive and four proactive domains. For brevity the framework below provides abstractions without examples. Detailed examples can be found in Dove (2001, 2012), with general coverage of case-making in Sillitto (2013). Note that response requirements are system operational-time requirements, not system design-time requirements; and should be stated as operational needs independent of possible solution strategies which will evolve with time.

Proactive domains:

- Creation/Elimination—what range of opportunistic situations will need modules assembled into responsive system configurations; what elements must the system create during operation that can be facilitated by modules and module pools; what situational evolution will cause obsolesce of modules which should be removed?
- Improvement—what improvements in system response performance will be expected over the system operational life?
- Migration—what evolving technologies and opportunities might require future changes to the infrastructure?
- Modification—what evolving technologies, opportunities, and situations might require future modifications to modules?

Reactive domains

- Correction—what types of response activities might fail and need correction?
- Variation—what operational conditions and resources vary over what range when response capabilities must be assembled?
- Expansion/Contraction—what are the upper and lower bounds of response capacity needs?
- Reconfiguration—what types of situations will require modular system reconfiguration to respond effectively?

Concluding Remarks

So called fourth-generation warfare is characterized as relatively small guerilla group activity, and fifth generation warfare is characterized as super-empowered individuals. Both leverage newly affordable technology to non-conventional advantage—and though warfare conjures up attacks on nation states by nation states, warfare encompasses undesirable intervention directed at any entity, from a concerted effort to gain advantage on a commercial competitor to an individual's revenge against a perceived injustice. But affordability is relative. With high stakes, the resources applied by organized crime and nation states are less constrained, with access to the brightest of minds and the most sophisticated technologies.

Fielding sustainably secure systems today is critical to system mission needs, yet difficult when system security is less than a paramount thoughtful concern of the system engineering processes. Responsibility lies with both acquirer, to demand it, and supplier, to provide it even when not demanded. The acquirer must place responsibility for system security on the systems engineering activity. The supplier must enable sustainable security and enable agile lifecycle security processes throughout the operational lifetime.

References

Dove, R. 2001. The Language, Structure, and Culture of Agile Enterprise. Wiley.

Dove, R. 2012. Agile Systems and Processes: Necessary and Sufficient Fundamental Architecture (Agile 101). INCOSE Webinar, 19 September. www.parshift.com/s/AgileSystems-101.pdf.

Kushner, D. 2013. The Real Story of Stuxnet. IEEE Spectrum, March. http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet.

Nagel, R. N. 21st Century Manufacturing Enterprise Strategy Report. 1992. AMEF N0001-92 Prepared for the Office of Naval Research. www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA25703 2.

Sillitto, H. G. 2013. Composable Capability – Principles, Strategies and Methods for Capability Systems Engineering. INCOSE IS13 International Symposium, INCOSE, Philadelphia, PA, 24-27 June.