**OpenC2 Language Security Outline**

**Introduction / Executive Summary**

* Focus of OpenC2 is the language, not the ACD system
* Introduce concept that a poorly designed/implemented OpenC2 may enrich an environment for potential adversaries
* Objective of OpenC2 should be to provide rapid response while not introducing new avenues of exploitation

**Threat Landscape**

* Control planes of all systems/networks is a primary target (think keys to the kingdom) since control planes are often the enablers of access.
* CND systems are also key targets as part of subverting access controls and detection.
* Many attacks focus on protocol/language exploitation based on lax enforcement and input validation.

**Operating Environment**

* Mixed environments.
* Mixed device smarts.
* Inheritance of legacy devices and architectures.
* Varied regulatory and compliance landscape leads to varied design choices.
* The OpenC2 language should be thin and flexible enough to fit within a wide range of environments.

**Security Implications of Language Structure**

* Only core security elements should be added to syntax
  + applicable to all targets and actuators
  + applicable to all contexts
* Context-specific security elements as specifiers
  + ignored by devices that don't have the required capability
  + not mandatory in all domains
* Security analysis must account for possibility of specifier being ignored
* Enduring, enclave-wide context-specific security requirements can be articulated as policy, rather than sent with each command
* Hooks for future development

**CIA and AAA**

* Availability - if the message does not reach the destination, then the active part of ACD is removed.
* Integrity - messages should not be altered in stream; ability to detect/resist replay, MITM.
* Confidentiality – environment considerations (Peer-to-Peer vs. centralized) and localized vs. shared environments;
* Authentication, Authorization and Accounting
  + Authentication: does end device authenticate origin of message?
  + Authorization: does end device track authorized orchestrators or peers? What about authorized actions (see verb white listing)?
  + Accounting: logging of all actions/outcomes; see section below regarding architecture and out-of-band mgmt.

**Verb White Listing**

* More advanced target devices (firewalls, IPS, etc…) could provide configuration options to support a white list of approved verbs and orchestrators.
* Capability to drop/reject commands from un-approved sources.
* The nature of changing a system’s operating posture requires vigilant consideration to type enforcement.

**Architecture Considerations**

* Consideration for the Orchestrator operating on trusted OS.
* Potential impact of containers/dockers.
* Peer-to-Peer vs. Centralized.
* Out-of-band Management, Out-of-band Monitoring (control plane vs. data plane)
* The Network Security Monitoring system (NSM) should be configured to receive copies of all published instructions from the orchestrator and executed commands from the end devices. If these two inputs do not match, then a possible compromise of the infrastructure should be investigated.
* Assuring end-to-end authentication/authorization in hierarchical architecture
  + entity to be authenticated only changes if command is changes (specified) by intermediate entities; command passed through from top-tier without change is attributed to top tier; command specified at lower echelon is attributed to specifying echelon -- if higher order source needs to be conveyed, it must be articulated as a specifier

**Security Impacts on Performance**

* Resources required to process additional layers of security (CPU, RAM, etc…)
* Key mgmt. – introduces additional complexity and mgmt overhead.
* Vendor integration (not all vendors parse PKI fields the same)
  + Example: Subject Alternative Name parsing varies by vendor.

**Other concerns**

* How to integrate modifications to infrastructure with Configuration Management?
* Should forethought be given to potential impacts of a pre-defined set of actions/changes permitted by the ACD system? Examples might include PII, PHI, PCI-DSS and other regulatory guidelines.
* A data flow with potential down-stream impacts may be required to validate permitted ACD actions at limited set of infrastructure points.
* Integration with existing security suite: It is possible the orchestrator is out of sync with the SIEM and detects/blocks out flows which are not part of establish network baseline. Alerts based on issued actions may need to be sent to SIEM for situational awareness.
* JSON or XML – ripe for exploitation
  + JSON concerns: eval() function. If input validation is not performed, then JSON becomes an enabler for adversaries with access to the system.
* Protocol implementation – TCP vs. UDP vs. other.
  + The choice of protocol implementation will have a direct impact on the potential security of the communication channel between the orchestrator and end points.
  + While UDP offers speed, the downsides are Availability and Security.
  + UDP tends to increase risks related to replay, spoofing.
  + UDP does not natively provide message ordering or congestion control.
  + Performance testing required to validate broad assumptions regarding performance.