**OpenC2 I.A.**

**Implementation Considerations**

**Introduction**

* Emphasis placed on separation of problem space and solution space
* 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 lack of 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 essential and core security elements should be added to syntax
  + desired effect not achievable using message stack
  + 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
* Thoughtful distinction between the payload syntax and the message wrapper

**Security Topics**

**Authentication**

Authentication is the process of determining whether someone or something in the context of OpenC2 is in fact who or what it is declared to be. In the overall operations of OpenC2 in context of machine-to-machine, the systems need to securely authenticate to verify it is the authorized system involved in any interaction and not a rogue entity. With the increasing number of Internet-enabled devices, reliable machine authentication is crucial to allow secure communication in automated network environments. In the Internet of things (LoT) scenario, almost any imaginable entity or object may be made addressable and able to exchange data over the network. It is important to realize that each access point is a potential intrusion point. Each network device that has an OpenC2 component needs strong machine authentication.

**Authorization**

Following authentication, a user must gain authorization for doing certain tasks. After logging on to a component, such as an orchestrator, the user account may try to issue actions. The authorization process should be in place to determine whether the user has the authority to issue such actions. Authorization is the process of enforcing policies: determining what types of qualities of activities, resources, or services/actions a user is permitted. Usually authorization occurs within the context of authentication. Once a user has been authenticated, they may be authorized for different types of actions depending on the policy assigned. In the context of OpenC2, this policy enforcement is necessary because even though a certain device can execute any action it is tasked to perform, some actions should not be executed at only certain times, or not at all as they could lead to network or device compromise.

There are actions within the OpenC2 language that can be grouped by their general activity. Each group of actions may need to have some level of authorization to allow such actions to be performed. One set of actions that control permissions and accesses are a desired area of interest of an attacker that can use these actions for his advantages. The OpenC2 commands are DENY, CONTAIN, ALLOW, which either of these used in the wrong context would lead to network compromise. Based on the security issues with authentication, the same concerns are with command authorization. OpenC2 syntax does not have inherited user schemas or even user to command mapped permissions that allow or not allow the ACTION to be performed.

Consider the following:

STOP (

[target (type=process, [id=123345])],

[actuator (type=endpoint ,[id=9876t6])],

[immediate]

)

The STOP construct would stop a process on an endpoint with an immediate action when the message is received. If this was a mission critical device that received the command without authorization, the network could be rendered inoperable.

**Accounting**

Accounting will be necessary for a multitude of activity to improve communication paths and actions within an OpenC2 environment. Accounting will provide the ability to measure resources a user or system component i.e. Orchestrator, consumes during access. This could include the amount of system time or amount of messages has sent or received during a session. Accounting is carried out by logging of session statistics and usage information and is used for authorization control, trend analysis, resource utilization, performance, and capacity planning. Overall all of these are important data captures to improve the configuration and deployment of OpenC2 components.

**Auditing**

OpenC2 is the ability to audit the activity that comes from the while maintain low latency and minimal overhead.

Actions and the following results that are the direct result of OpenC2 should be recorded and analyzed for security areas such as forensics, secure implementation, security architecture of impact changes within the environment, and completion of such tasks.  Currently there are no enforcements within the OpenC2 framework that would directly accomplish such actions.

**Non-Repudiation**

The current OpenC2 syntax only supports sending and receiving messages but does not have any mechanisms to ensure the transferred message has been sent and received by the parties claiming to have sent and received the message. Nonrepudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

**Integrity / Man in the Middle**

Currently there is no message protection when using the OpenC2 syntax. When a message is received by an OpenC2 actuator, it should ask two questions: whether I trust the sender (another actuator or orchestrator) and whether it created this message. Assuming that the sender trust has been established one way or another, the target has to be assured that the message it is looking at was indeed issued by the server, and not altered along the way

**Confidentiality**

Confidentiality protections may apply to the entire message being processed, or only to certain parts of it.

Applying encryption requires conducting an extensive setup work, since the communicating parties now have to be aware of which keys they can trust, deal with certificate and key validation, and know which keys should be used for communication.

**Replay**

Even a valid message may present a danger if it is utilized in a "replay attack". i.e. it is sent multiple times to the server to make it repeat the requested operation. This may be achieved by capturing an entire message, even if it is sufficiently protected against tampering, since it is the message itself that is used for attack now. Usual means to protect against replayed messages is either using unique identifiers (nonces) on messages and keeping track of processed ones, or using a relatively short validity time window.

**Verb Whitelisting**

Many devices such as a firewall are capable to accept commands such as DELETE, MODIFY, ACCEPT, DENY, START, STOP, RESTART however even if such a device is capable of accepting the commands and act upon an action, not all actions should take place especially during production operating hours. Actuators and Target devices will blindly act upon actions, in essence OpenC2 command hierarchy is too much of an enabling system without certain capability checks in place.

**Security Impacts on Performance**

Resources required to process additional layers of security (CPU, RAM, time, etc…)

Key management introduces additional complexity

Vendor integration challenges (Example: Subject Alternative Name parsing varies by vendor)

Modifications to the OpenC2 language structure in support of complex tasks such as encryption should be kept to a minimum to avoid re-inventing already established standards and taxing devices which might not be capable of the required processing overhead. This is an area where the deployed environment can best leverage existing and compatible capabilities to perform such services.

**Architecture / Implementation Considerations**

*Peer-to-Peer*

Peer-to-peer architectures can present many atypical issues not found on other domain-based systems. Secure connections have different KMI challenges and basic issue of trust must be addressed differently.

*(Multi) Hub and Spoke*

*Strict Type Enforcement*

Strict type enforcement and the related input validation may prove challenging with the current flexible structure (everything is optional).

*Integration with Configuration Management*

The tracking (auditing) and reporting of changes initiated by the OpenC2 ecosystem needs to feed back to the CM solution as a matter of both best practice and regulatory and certification & accreditation compliance.

*Out of Band Management (OOBM)*

OOBM is a common best practice with renewed focus based on the threat landscape listed earlier in this document. C2 systems are prime objectives for bad actors and OOBM offers another layer in the defense-in-depth model.

*Domain Federation*

A federation trust model is often found in partner organizations with explicit trust and a shared security and/or accreditation boundary.

*Certification and Accreditation (C&A) – Regulatory Compliance*

Many environments will face a range of regulatory compliance or certification and accreditation requirements. A combination of the native OpenC2 language feature set and deployed architecture must address fundamental security features (auditing, CIA, AAA, encryption) in order to achieve compliance or Authority to Operate (ATO).