**OpenC2 I.A.**

**Implementation Considerations**

**Background**

Referenced from DARPA - U.S. military, government and commercial IT networks face constant cyberattack from both criminal and state-sponsored adversaries. Current IT security response practices to these attacks boil down to four steps: find the invading code, unplug the affected systems, create security patches to thwart particular attacks, and apply those patches network-wide. This reactive engagement model is effective on a case-by-case basis but does not address key advantages attackers have—for example, adversaries can easily make small changes to malware that bypass patches and distribute that new malware on a massive scale. To stay ahead of increasingly sophisticated, stealthy and dangerous threats, defenders must move beyond traditional static defenses to exploit the natural advantages of their IT systems and expertise.

DARPA’s Active Cyber Defense (ACD) program is designed to help reverse the existing imbalance by providing cyber defenders a “home field” advantage: the ability to perform defensive operations that involve direct engagement with sophisticated adversaries in DoD-controlled cyberspace. Created in December 2012, the program seeks to develop a collection of synchronized, real-time capabilities to discover, define, analyze and mitigate cyber threats and vulnerabilities. These new proactive capabilities would enable cyber defenders to more readily disrupt and neutralize cyberattacks as they happen. These capabilities would be solely defensive in nature; the ACD program specifically excludes research into cyber offense capabilities.

OpenC2 has been a concept to be used in junction with the ACD program. In context with ACD, an objective is to define a set of universally understood Integrated Defensive Cyber Operations (IDCO) Command and Control (C2) commands and a command structure that can be contextualized for specific environments. The universally understood command set, OpenC2, will cover the range of common actions expressed at a high level. These high level commands are then interpreted in terms of the context, the concerns and assets of the entity executing the commands. These interpretations need to be commonly understood locally, but not globally. They are only meaningful in the context in which they are used, so cross-context adjudication is not required. Each enterprise or enclave provides its own implementation via the specific sensors and actuators available, and so can specify the common command set to reflect the implementations of which it is capable.

Preface:

Achieving adequate information security for organizations, mission/business processes, and information systems is a multifaceted undertaking that requires:

* Clearly articulated security requirements and security specifications;
* Well-designed and well-built information technology products based on state-of-the-practice hardware, firmware, and software development processes;
* Sound systems/security engineering principles and practices to effectively integrate information technology products into organizational information systems;
* Sound security practices that are well documented and seamlessly integrated into the training requirements and daily routines of organizational personnel with security responsibilities;
* Continuous monitoring of organizations and information systems to determine the ongoing effectiveness of deployed security controls, changes in information systems and environments of operation, and compliance with legislation, directives, policies, and standards and
* Information security planning and system development life cycle management.

From an engineering viewpoint, information security is just one of many required operational capabilities for information systems that support organizational mission/business processes— capabilities that must be funded by organizations throughout the system development life cycle in order to achieve mission/business success.

**Introduction**

* Emphasis placed on separation of problem space and solution space in context to OpenC2 implementations.
* Focus of OpenC2 as 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

**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. There are many challenges to find the right authentication model that can support a machine-to-machine communication method at extremely fast rates of operating.

**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 communication 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 and a verification that intended operations are working as intended.

**Auditing**

Audit trails are necessary in machine-to-machine communications. Audit trails maintain a record of system activity by system or application processes and by user activity. In conjunction with appropriate tools and procedures, audit trails can provide a means to help accomplish several security-related objectives, including individual accountability, reconstruction of events, intrusion detection, and problem identification. In audit terms, such activities are often called events, and auditing OpenC2 functions could be called event logging. Typical events include:

* Logins (successful, unsuccessful)
* Logouts
* Remote System Accesses
* System Application manipulation (start, stop, restart, change / modified status)
* File Opens, closes, renames, deletions, modifications.
* Changes in privileges or security attributes (e.g. a change in a network service label or a user’s permission)

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.  This type of auditing might prevent security violations from occurring. Currently there are no enforcements within the OpenC2 framework that would directly accomplish such actions.

**Non-Repudiation**

Nonrepudiation is the assurance that someone cannot deny something. Typically, nonrepudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

A repudiation attack happens when an application or system does not adopt controls to properly track and log users' actions, thus permitting malicious manipulation or forging the identification of new actions. This attack can be used to change the authoring information of actions executed by a malicious user in order to log wrong data to log files. Its usage can be extended to general data manipulation in the name of others, in a similar manner as spoofing mail messages. If this attack takes place, the data stored on log files can be considered invalid or misleading.

Managerial control of enforcing confidentiality throughout an OpenC2 deployment architecture along with the implementation control considerations of adding encryption could be possible to enhance the operational effectiveness of OpenC2 usage and provide a level of technical assurance of message authenticity.

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.

**Integrity**

Integrity of the messages ensures that there is some form of proof that information has not been modified either accidentally or deliberately by non-authorized entities. Integrity insures that all resources in the system are always available in a “high availability” form i.e. that the content, meaning, structure and function of each resource are always available and preserved in its correct form. It should be noted that the definition of integrity also implies availability.

Without message integrity there are several security risks that are prevalent and currently used by attackers, applicable to OpenC2, which include:

* Man in the Middle – ability to intercept the messages while in transit and possible use the information gathered for reconnaissance, and message tampering
* Forgery of the message – changing the content of the message so a different action is received.
* Replay attack - 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.

**Confidentiality**

Confidentiality means that the contents of the information processed, stored, or transmitted are protected. That means the existence of a piece of information , its sources, its recipients, its contents, the meaning of structure of system resources dealing with the piece of information (message) must not be made available or disclosed in any way to unauthorized entities. 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. (to be discussed in another section)

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

**Verb Whitelisting**

Even though devices have support for a variety of methods of configuration management and operational change configurations, the ability to use such methods at any given time and at any given moment would be considered hazardous to network and device operations. It would be an operational method to form a whitelist of device operational commands and have a technical consideration of allowing certain commands to be used by certain operators and at different times of execution (off-hours, maintenance windows). An implementation model could be developed with the commands and OpenC2 syntax to make best use of creating a best use model of operating the devices autonomously.

For example, 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.

**Architecture / Implementation Considerations (section yet to be completed)**

*Peer-to-Peer*

Peer-to-peer architectures can present many atypical issues not found on other domain-based systems. Secure connections have different key management interface 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 is a requirement. Within an OpenC2 implementation, it must be possible to receive any arbitrary string of bytes and determine if it is a valid OpenC2 command. As long as all the elements in the message are validated against a controlled set of vocabulary, using optional modifiers would not be a problem.

*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).