An Open Source Command and Control Language for Resilient Cyber Systems

Pat Muoio
Director of Research and Development
G2, Inc.

November 2015

The Objective

- Define a lightweight language for efficient communication of commands to support near real-time response.
- Responses can be orchestrated within or among enclaves

Assumptions

- Compute environments will be heterogeneous and differences among enclaves will be significant
- There will never be complete agreement on a universal ontology to underpin meaning across enclaves
- Machines are commanded, not persuaded
  - Response time is short
  - Nuance, rationale, motive don’t matter

OpenC2 Language Preliminary Requirements

- Language must be lightweight for efficient machine-to-machine communication of cyber response action.
- Core language must be small and extensible to address the requirements of an operational context.
- Language must say what to do, not how to do it.
- Support effects-based with specific approach determined by environment/context
- Must to be able to command a specific actor, a list of actors, or a class of actors.

OpenC2 Characteristics

- Support cyber relevant time C2 coordination and response actions, compact and low latency
- C2 language is independent of infrastructure/architecture or message format
- C2 language permits different levels of abstraction
  - Language supports commands that are either tasking/response actions or notifications
    - tasking changes the state
    - notifications require some level of sense making/analytics
  - universally understood or context/environment specific
  - specifiable to task different types of actuators or specific actuators (e.g., sensor, endpoint, network device, human) even as the types of actuators grow (e.g., Internet of Things)
- C2 language can accommodate variability in security context, acknowledgement scheme, logic, workflow, and synchronization

OpenC2 Syntax

- <ACTION> (
  [target (type=<TARGET_TYPE>, *specifiers*)],
  [actuator (type=<ACTUATOR_TYPE>, *specifiers*)],
  [*modifiers*]
)
  - ACTION
    - gathers and conveys information, controls activities and devices, controls permissions and access
  - TARGET is the object of the ACTION (optional)
    - e.g., device, connection, person, process, network, data
  - ACTUATOR performs the ACTION (optional)
  - *specifiers* further identify TARGET(s) and ACTUATOR(s), individuals or groups
    - *accommodates hierarchies, inheritance, lists*
  - *modifiers* provide additional information for the ACTION (optional)
    - e.g., time, frequency, degree/extent, priority, location

The “Universal” Terms

- High level of generality to remain relevant across enclaves
- Verbs
  - Scan, locate, acknowledge, query, start, stop, restart, pause, resume, set, update, move, remove, modify, execute, deny, allow
- Nouns (both subject and object)
  - Device, person, process, network, data, parameter
- Context Neutral Modifiers
  - Deadline, duration, periodicity, frequency, degree, priority, location

Specifiers

- Each noun can be specified;
  - Meaning of the specifier is determined by the noun modified
- Examples
  - Device(type)(brand)(instance)
    - Verb Device (firewall)… (meaning whatever firewalls you have that can do this)
    - Verb Device(firewall)(brandX) (1, 2, 3)… (meaning those 3 particular brand X firewalls)
    - Verb Device(firewall) (brandY) … (meaning all brand Y firewalls)
    - Verb Device (firewall) (1,2,3) … (meaning firewall 1,2, and 3 regardless of brand)
- Absence of follow-on modifier means you do for all at that level of generality
- Allow lists and accommodate hierarchies in specifiers
- Enable Inheritance

OpenC2 Syntax Flexibility

ACTION	TARGET	SPECIFIER	ACTUATOR	SPECIFIER	METHOD

- Effects-based (no actuator specified)

DENY	ip	id=<IP_ADDR>			

- Specify actuator type

DENY	ip	id=<IP_ADDR>	network		

- Hierarchical Types

DENY	ip	id=<IP_ADDR>	network.router		

- Further specify actuator

DENY	ip	id=<IP_ADDR>	network.router	id=<BGPRTR>	

- Further specify action

DENY	ip	id=<IP_ADDR>	network.router	id=<BGPRTR>	BGP_Blk_Hole

A Key Distinction

Inter-enclave commands

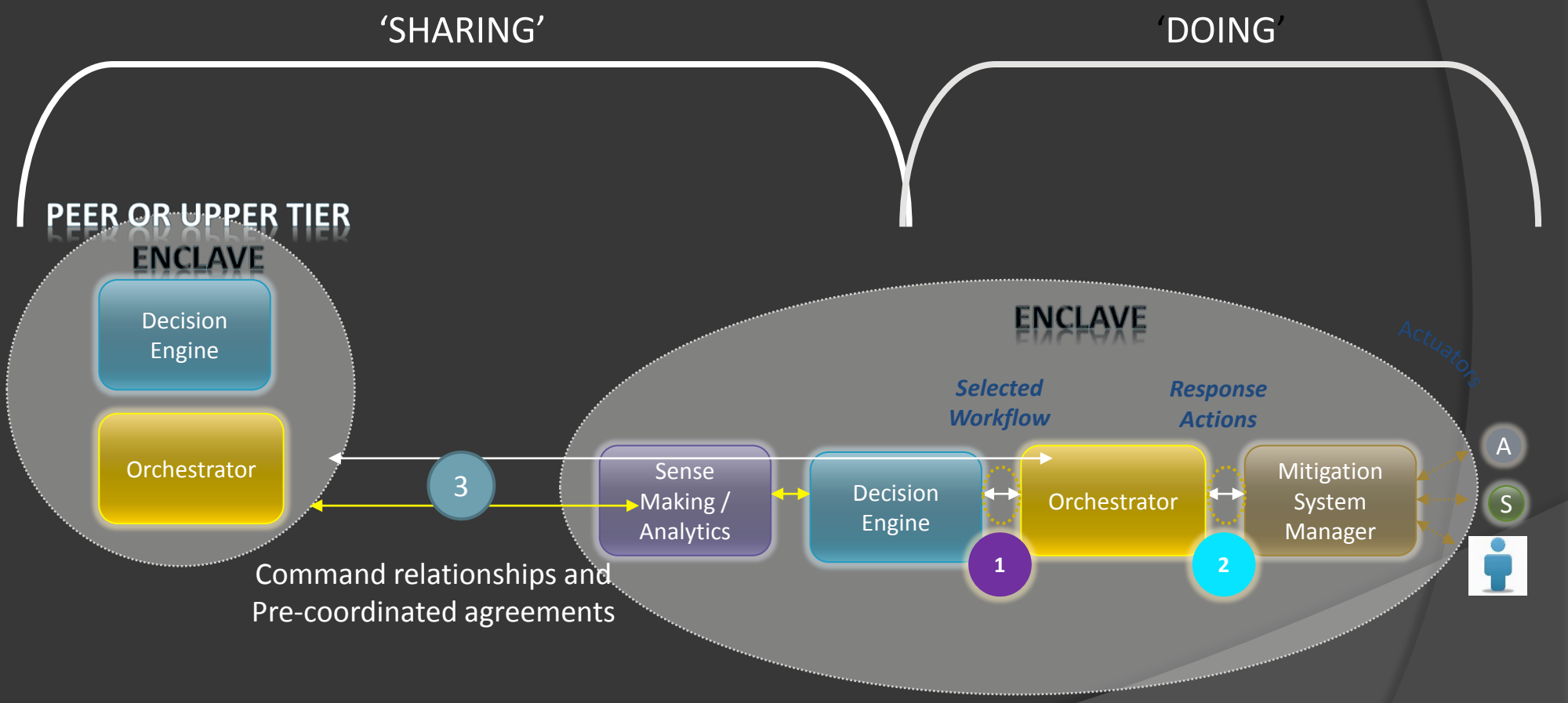
vs.

Sharing suggested Courses of Action

Inter-Enclave Commands

- Require established command hierarchy
  - In general, hierarchical commands are meant to be obeyed no questions asked
  - Enclaves are governed by the same policy
- Or pre-coordinated command agreement
  - E.g., supply chain, members of conglomerate
  - Agreement could stipulate level of self-determination allowable to receiving enclave

Inter-enclave commands

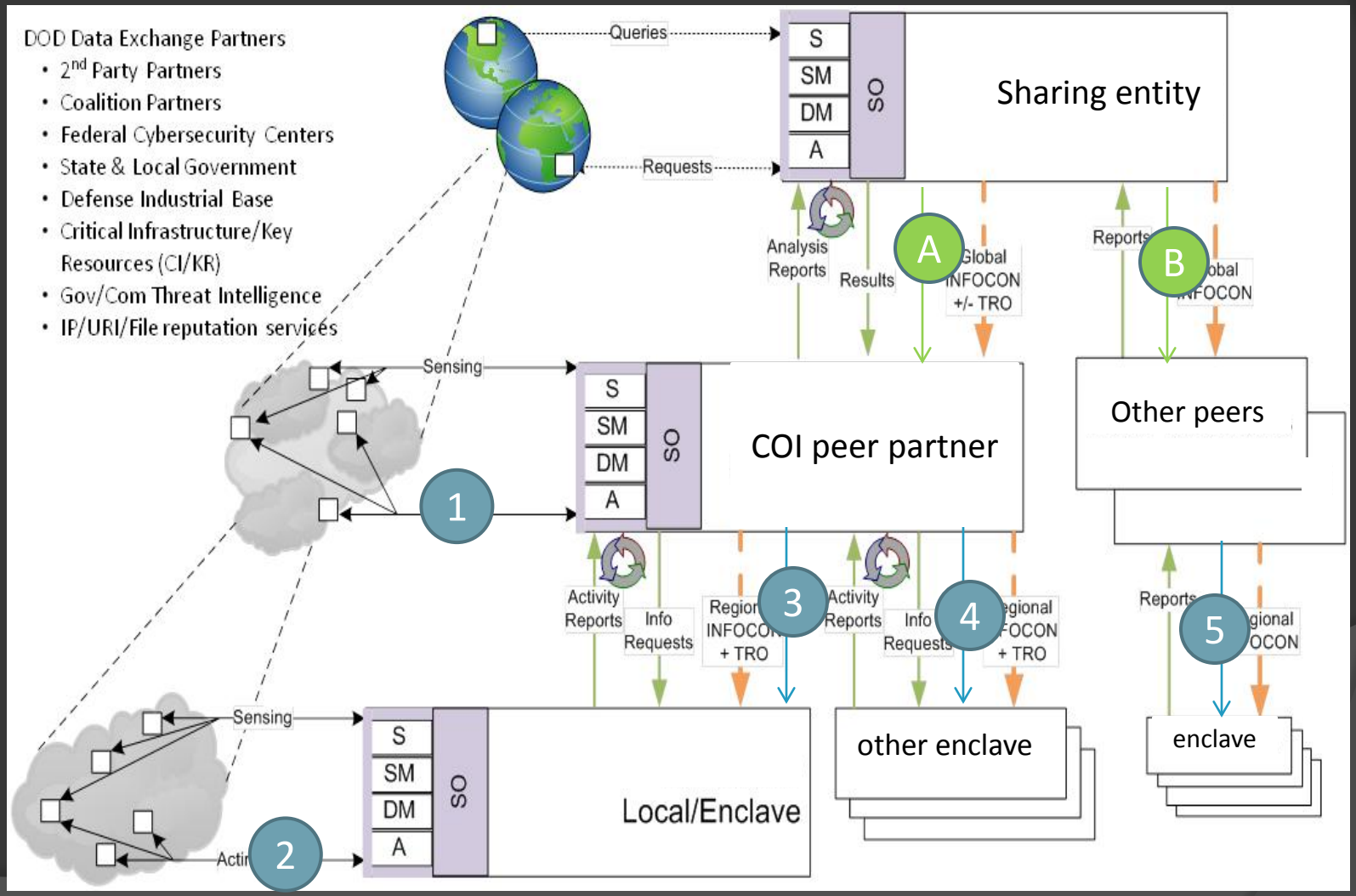


1, 2 intra-enclave OpenC2 commands
3 inter-enclave OpenC2 command

Sharing Suggested COAs

- Knowledge sharing activity designed to provide a response option as part of shared situational awareness
- Publishing entity has no authority over receiving entities
- Suggested actions are taken under advisement
- Receiving entity might not have the capacity for action assumed in the command
- Context is relevant to the recipient
  - Statements about confidence levels, provenance, specifics of triggering threat, trustworthiness of source all could be relevant to receiving entity's decision to act on the information
- Using the OpenC2 syntax to express the COA could provide clarity and brevity, but the exchange needs to include more than just the command

Sharing suggested COAs



A, B Suggested COA

1, 2, 3, 4, 5 OpenC2 Commands