Open Command and Control (OpenC2) Profile for Firewall Functions

Version pre-release 0.1

24 April 2017

TABLE OF CONTENTS

1 Introduction 3

1.1 Purpose 3
1.2 Scope 3

2 openc2 language binding 5

2.1 Actions 5
2.2 Responses and Alerts 5
2.3 Data Modeling 6
2.3.1 Target and Target Specifier Data Model 6
2.3.2 Firewall Specifier Data Model 7
2.3.3 Modifier Data Model 8

3 sample openc2 commands 9

3.1 Deny and Allow 9
3.2 Set 11
3.3 Update 12

List of Tables

Table 2.1-1: Firewall Actions 5
Table 2.2-1: Response codes applicable to all actions 6
Table 2.3-1: Target data model applicable to Firewalls 6
Table 2.3-2: Firewall Specifiers 7
Table 2.3-3: Firewall Modifiers 8

1 INTRODUCTION

Firewalls use rules to control incoming and outgoing traffic. Essentially all networks will have one or more firewalls integrated within their cyber defense. First generation (also known as packet filter) firewalls may operate at line speed and are often deployed at the perimeter. Second generation (stateful) firewalls maintain a table of valid connections and reject packets that are not a part of a valid connection (typically TCP). Third generation (application layer) firewalls perform some level of inspection within packets/flows to determine if the payload is in fact consistent with expected content for a given application. Firewall functionality may be provided by single purpose devices or may be provided as a function for a multi-purpose device or a system.

This paper outlines the set of actions, targets, specifiers and modifiers that integrates first generation firewall functionality with the Open Command and Control (OpenC2) command set. Through this command set, cyber security orchestrators may gain visibility and provide control into the firewall functionality in a manner that is independent of the vendor or generation of the firewall. In the context of this document, 'firewall' refers to the first generation packet filter firewalls. Next generation firewall products that provide additional functionality are referred to other actuator profiles. NEED TO DEFINE

1.1 Purpose

The purpose of this document is to:

- Identify the OpenC2 ACTIONS that are applicable to the actuators with firewall functionality.
- Identify the TARGETS and TARGET specifiers that are applicable to the firewall class of actuators.
- Identify SPECIFIERS and MODIFIERS that are applicable and/or unique to the firewall class of actuators
- Provide sample OpenC2 commands to a firewall

All components, devices and systems that provide firewall functionality will implement the ACTIONS, TARGETS, SPECIFIERS and MODIFIERS identified as minimum to implement (MTI) in this document. Actions that are applicable, but not necessarily required for firewalls will be identified as optional.

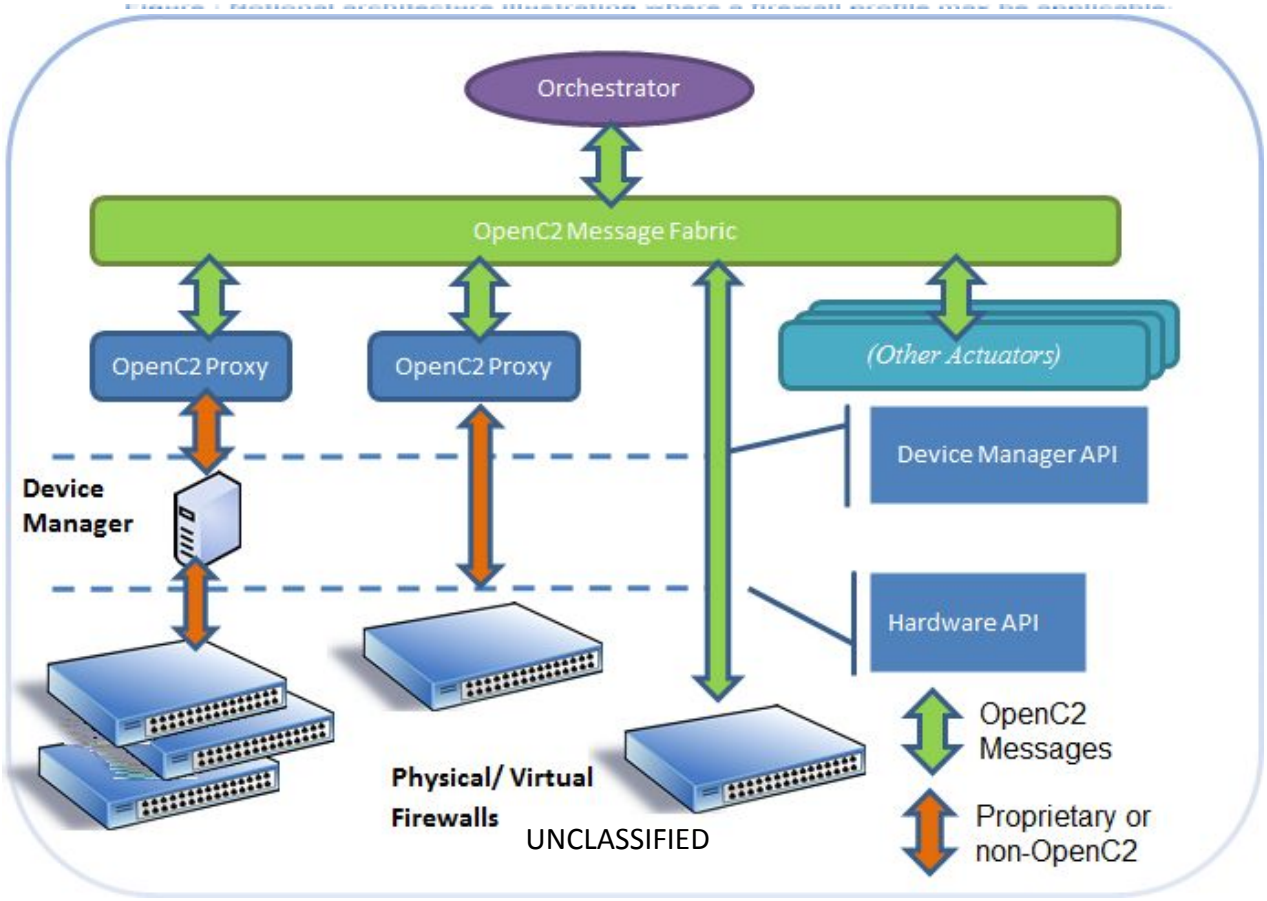
1.2 Scope

Figure 1 presents a notional OpenC2 implementation which illustrates cases were a firewall profile may be required and the components within the network that may interact with or be affected by OpenC2.

- OpenC2 message fabric: The transport mechanism for passing OpenC2 commands between OpenC2 compliant entities within the network.
- Orchestrator: Products that send commands, receives responses, and manages the execution of a course of action involving one or more actuators. The orchestrator needs a priori knowledge of which commands the actuator can process and execute therefore must implement the

profiles for any device that it intends to command.

- OpenC2 Proxy: An abstraction of the firewall functionality that maps (or translates) OpenC2 commands to an appropriate API (i.e. a mitigation manager or vendor API).
- Device Manager: Interfaces with one or more physical or virtual firewalls. A device manager is a means to provide mitigation system management, which includes participation in OpenC2 workflow processing, transforming actions into a format suitable for a given device, set of devices or capability within a device that provides the firewall functionality. A proxy between the OpenC2 message fabric and the device manager may implement the firewall profile and map the commands to the device manager's API or the device manager itself may natively support OpenC2 at its API (thus removing the need for the proxy).
- Proxy to Physical or virtual firewalls. An OpenC2 proxy must implement the firewall profile and map the commands to the vendor API.
- Native OpenC2 support: In the future, there may be devices that natively support OpenC2, and will be required to implement the firewall profile.
- Other Actuators: A product may provide multiple cyber defense mechanisms including firewall functionality as a subset of its capabilities thus the firewall profile is (in addition to other profiles) needed.


Orchestrator
OpenC2 Message Fabric
OpenC2 Proxy
OpenC2 Proxy
(Other Actuators)
Device Manager API
Device Manager
Hardware API
Physical/ Virtual Firewalls
OpenC2 Messages
Proprietary or non-OpenC2

2 OPENC2 LANGUAGE BINDING

The OpenC2 Language Specification (OpenC2 Forum, 2016) describes a vocabulary by which network elements may be commanded and controlled. By design, the language is intended to convey high level actions so that the desired effects of a command can be communicated despite a lack of detailed knowledge of the cyber defense components. As additional information is made available, the language is extensible with context specific modifiers and specifiers which permits additional granularity of the command. In this section the minimum to implement (MTI) and Optional Actions, Targets, Specifiers and Modifiers that are appropriate for the firewall class of actuators are identified.

2.1 Actions

All OpenC2 commands require an Action. Table 2-2 summarizes the set of actions that are meaningful in the context of firewalls. Eight OpenC2 actions are identified as MTI that must be implemented for actuators that provide firewall functionality. Four OpenC2 actions are meaningful in the context of firewalls but are considered optional. Section 3 provides examples for each action in the context of a target and its specifiers. Sample use cases are provided at (need url for wherever we store use cases)

OpenC2 actions are persistent in firewalls unless they are explicitly identified as running or temporary through the use of modifiers.

Table 2.1-1: Firewall Actions		
query	The *query* action initiates a single request for information. Used to determine the state or settings of the firewall.	MTI
deny	The *deny* action is used to prevent a traffic from reaching a destination or preventing access.	MTI
allow	The *allow* action permits traffic or access.	MTI
set	The *set* action changes a value within a configuration, a setting or some other value	MTI
notify	The *notify* action is used to direct an entity to send alerts to another entity. Notify is not used for normal operations, but is required for the use case where you need to rapidly direct the firewall to send alerts.	MTI
update	The *update* action instructs the component to retrieve and process a software update, reconfiguration, or some other update.	MTI
delete	Removes data, files and or entries.	MTI
save	Commits data or state to memory.	MTI
start	The *start* action initiates a process, application, system or some other activity.	Optional
stop	The *stop* action halts a system or ends an activity.	Optional
restart	The *restart* action conducts a *stop* of a system or an activity followed by a *start*.	Optional
redirect	In the context of a firewall, redirect is used to forward the packet, traffic or flow to another actuator or honey net and not necessarily to induce an alternate path to the destination.	Optional

2.2 Responses and Alerts

Response and Alerts messages originate from the actuator and are informative rather than a command or request that the recipient execute some task(s).

Alert is used to signal the occurrence of an event and are unsolicited. Response is used to provide any data requested as a result of an action such as acknowledgement, providing the status, or additional information related to the requested action. The recipient of the response can be the original requester of the action or to another recipient(s) designated in the modifier of the action.

Status codes apply to all actions and are presented in table 2.2. The formats and what is expected in the response and alerts for specific actions will be captured when the specifiers for the various actions are defined in section 2.3 and examples will be provided in section 3. Responses and Alerts associated with MTI actions MUST be implemented. Implementations that include optional actions MUST implement the Responses associated with the implemented action.

Table 2.2-1: Response codes applicable to all actions

Status Code	Status Text	MTI
102	Processing. Command received but action not necessarily complete	MTI
200	OK	MTI
301	Cannot execute, redirect to another entity. In this response code, a specifier that indicates a suggested entity MUST be included in the data field.	Optional
400	Unable to process command, parsing error	MTI
401	Authentication or authorization failure	MTI
403	Forbidden	MTI
500	Server Error	MTI
501	Not implemented	MTI

2.3 Data Modeling

Applications that support OpenC2 MUST produce and accept messages that are valid according to the schema shown in Appendix A. Such applications MAY support additional messages not defined in the schema.

2.3.1 Target and Target Specifier Data Model

The TARGET is the object of the ACTION (or alternatively, the ACTION is performed on the TARGET). This document will use the OpenC2 schema which was derived from the STIX data model for cyber

observables. Table 2.3.1 lists the TARGET namespace that are applicable to firewall functionality.

Table 2.3-1: Target data model applicable to Firewalls

Target	Description/ Notes	Applicable Actions	MTI
domain-name	Fully qualified domain name. Supported by some firewalls. The actual rule within the firewall will be an IP address so the DNS resolution must take place. Most firewall implementations utilize tools that are optimized for DNS resolution.	deny, allow	Opt
Five-tuple	Consists of the address (source and destination), port number (source and destination) and protocol identifier. An 'incomplete' five-tuple may be sent to the firewall and it is left to the implementer's discretion with respect to the default value of the unspecified fields and/or if the unspecified fields make sense in the context of the device.	deny, allow	MTI
CIDR	In the interest of generating a more concise command, the use of the classless inter-domain routing notation to specific multiple addresses within the five-tuple is permitted.	deny, allow	Opt
Hostname	Supported by some firewalls. The actual rule within the firewall is an IP address so the hostname resolution must take place.	deny, allow	Opt
File	Typically supported by device managers but may be supported by some firewalls. The file object includes the path and name specifiers	Update	Opt
x-config	The x-config.fw object identifies configurable attributes of the firewall. The attributes include: • x-config.fw.logging (Boolean) •	Set	Opt

2.3.2 Firewall Specifier Data Model

An ACTUATOR is the entity that provides the functionality and performs the action. The ACTUATOR executes the ACTION on the TARGET. In the context of this profile, the actuator is the firewall and the presence of one or more specifiers further refine which actuator(s) shall execute the action.

The ACTUATOR is optional in an OpenC2 command. If absent, then any entity that can execute the action should execute the command. If the ACTUATOR field is specified, then only the entities identified to the degree specified in the actuator field act upon the command.

Whether or not an actuator specifier is meaningful in the context of a firewall is strongly dependent on the individual product, therefore are optional.

Table 2.3.2 identifies the specifiers that are applicable to the firewall actuator. Section 3 provides sample commands with the use of specifiers.

Table 2.3-2: Firewall Specifiers

Firewall Specifier	Description
Network	All network layer devices that implement the firewall profile
Perimeter	Perimeter; Firewalls with connections to the network and to external networks that maintain external routing tables
Internal	Internal; Firewalls that are not directly connected to devices external to the network.
Host	Hostname for a particular w/ firewall functionality
Ip-addr	Ip address for a particular w/ firewall functionality
Swid	Further specifies the Software ID for a particular device w/ firewall functionality
Named Group	User defined collection of devices with firewall functionality

There are firewall specifiers that only apply to a subset of firewall actions. These specifiers provide detail on how the action is executed.

Table 2.3-3: Action Specific Firewall Specifiers

Specifier	Type	Description	Applicable Actions	MTI
Drop	Boolean	Stop processing and do not send a notification to the source of the packet.	Deny	MTI
Reject	Boolean	Stop processing and send a notification to the source of the packet.	Deny	MTI
Complete	Boolean	Stop processing and send a false acknowledgment to the source that the processing was completed.	Deny	Optional
Running	Boolean	Any changes to a device are to be implemented as persistent changes. Setting the running modifier to TRUE results in a command that is not persistent in the event of a reboot or restart. The running modifier can be overridden by issuing a subsequent *save* action	Set, update,	Optional

2.3.3 Modifier Data Model

Modifiers provide additional information about the action such as time, periodicity, duration, location etc. Modifiers can denote the when, where, and how aspects of an action. Modifiers can be used to indicate whether the actuator should explicitly acknowledge receipt of the command, respond upon completion of the execution of the command, or provide some other status information. OpenC2 actions are persistent (or permanent) in their implementations. Running, non-persistent or temporary

commands can be achieved through the use of modifiers.

There are three sets of Modifiers:

- Universal; Applicable to all actions for all actuators. The universal modifiers are documented in the Language Description document and by definition apply actuators that provide firewall function.
- Action Specific; Applicable to specific actions for all actuators.
- Actuator Specific; Are only meaningful in the context of a particular actuator function.

Table 2.3.3 summarizes the Modifiers as they relate to firewall functionality.

Table 2.3-3: Firewall Modifiers

Modifier	Type	Description	Applicable Actions	MTI
id	string	The unique identifier for the action.	Universal	MTI
response	string	Indicate the type of response required for the action.	Universal	MTI
start-time	datetime	The specific date/time to initiate the action.	Universal	MTI
end-time	datetime	The specific date/time to end the action.	Universal	MTI
destination	String	Identifies where to send an acknowledgement or other response.	Universal	MTI

3 SAMPLE OPENC2 COMMANDS

This section will summarize and provide examples of OpenC2 commands as they pertain to firewalls. The sample commands will be encoded in verbose JSON, however other encodings are possible provided the command is validated against the schema presented in Appendix A. Examples of corresponding responses and/or alerts will be provided where appropriate.

The samples provided in this section are for illustrative purposes only and are not to be interpreted as operational examples for actual systems. Within the scope of this document, a # character indicates a comment, however it should be noted that OpenC2 itself does not support comments within a command.

3.1 Deny and Allow

Deny and allow are mandatory to implement and can be treated as mathematical complements of each other. Unless otherwise stated, the example targets, specifiers, modifiers and corresponding responses are applicable to both actions.

 # Block a particular connection within the domain and do not send a host unreachable

```
{"action": "deny",
        "target": {
```

```
                "type": “openc2:five-tuple",
                "specifiers": {
                        "Layer4Protocol": "UDP",
                        "ip-address-src”: 1.2.3.4
                        "ip-address-dst”: 1.2.3.5
                        "src-port": 10996
                        “dst-port”:443
                        }
                }
        "actuator": {
                "type": "openc2:firewall",
                "specifiers": {internal}
                },
        "modifiers": {
                {“id”:”UUID=123e4567-e89b-12d3-a456-426655440000”}
                {“openc2: drop”}
}
```

 # Block all ftp data transfers from hosts and request ack. Note that the five-tuple is incomplete

```
{"action": "deny",
        "target": {
                "type": “openc2:five-tuple",
                "specifiers": {
                        "Layer4Protocol": "TCP",
                        "src-port": 21
                        }
                }
        "actuator": {
                "type": "openc2:firewall",
                "specifiers": {endpoint},
                “options”:{ openc2: drop}
                },
```

"modifiers": {

{“id”:”UUID=123e4567-e89b-12d3-a456-426655440000”}

{response=TRUE}

}

 # Note that the response was requested and all endpoints that can execute the command should.

 # In this case, one of the endpoints successfully issued the deny but the endpoint located at 1.2.3.8 failed

{response

{Source: openc2:ip-addr=1.2.3.4}

{cmdref=123e4567-e89b-12d3-a456-426655440000 }

{statuscode=200}

}

{response

{Source: openc2:ip-addr=1.2.3.8}

{cmdref=123e4567-e89b-12d3-a456-426655440000 }

{statuscode=400}

}

 # Allow ftp data transfers to a particular ip address from any host. Note that the five-tuple is incomplete

{"action": "allow",

"target": {

"type": “openc2:five-tuple",

"specifiers": {

"Layer4Protocol": "TCP",

"ip-address-dst”: 1.2.3.5

"src-port": 21

}

}

"actuator": {

"type": "openc2:firewall",

"modifiers": {

{“id”:”UUID=123e4567-e89b-12d3-a456-426655440000”}

{response=TRUE}

#

{response

{Source: openc2:ip-addr=1.2.3.4}

{cmdref=123e4567-e89b-12d3-a456-426655440000 }

{statuscode=200}

}

3.2 Set

Set is mandatory to implement and is intended for toggling of values or enabling capabilities or functions. This action is distinct from *update* in that *set* needs to select from the allowed values of target (artifact, file, process, user-account, windows-registry-key) and the allowed actuator types. *Set* is intended for more atomic modifications rather than a full replacement of a configuration. In the context of firewalls, *set* is intended to modify data that impacts the firewall itself and it is inappropriate to use set to modify the ACL. Modifications of the ACL should utilize the *deny* and *allow* actions.

 # Turn the logging on for all of the network layer firewalls

{"action": "set",
"target": {
"x-config”: {“firewall.logging": true}},
"actuator": {
"type": "openc2:firewall": {
"named-group”: “network"}},

3.3 Update

Update is mandatory to implement and is intended for the device to process new configuration files, software updates, patches, policy updates etc. The *update* action is a compound action in that all of the steps required for a successful update (such as download the new file, install the file, reboot etc.) are implied. In the context of configuration updates, *update* is distinct from *set* in that a *set* command will provide the new value within the command itself while an *update* command may provide the location of the new files which are retrieved out of band.

 # instructs the firewalls to acquire a new configuration file. Note that all network based firewalls will install the new update because no particular firewall was identified. Host based firewalls will not act on this because network firewalls were identified as the actuator

{ "action": "update",
"target": {

```
            "file": {
                    "parent_directory": {
                            "path":"\\\\someshared-drive\\somedirectory\\configurations"},
                            "name": "firewallconfiguration.txt"}},
    "actuator": {
            "openc2:firewall": {
                     “named-group”:”network”
                            }
    }}
```

 # Instructs any firewall running a particular software load to install a software upgrade

```
{   "action": "update",
    "target": {
            "file": {
                    "parent_directory": {
                            "path": "\\\\someshared-drive\\somedirectory\\so"},
                            "name": " version2.2offirewallsoftware.exe"}},
    "actuator": {
            "openc2:firewall": {
                     "x-tagID": "firewallcompanyversion2.1"}
    }}
```