

I2NSF Working Group
Internet-Draft
Intended status: Informational
Expires: January 18, 2019

R. Kumar
A. Lohiya
Juniper Networks
D. Qi
Bloomberg
N. Bitar
S. Palislamovic
Nokia
L. Xia
Huawei
J. Jeong
Sungkyunkwan University
July 17, 2018

Information Model for Consumer-Facing Interface to Security Controller
draft-kumar-i2nsf-client-facing-interface-im-07

Abstract

This document defines an information model for Consumer-Facing interface to Security Controller based on the requirements identified in [[I-D.ietf-i2nsf-client-facing-interface-req](#)]. The information model defines various managed objects and relationship among these objects needed to build the interface. The information model is organized based on the "Event-Condition-Event" (ECA) policy model defined by a capability information model for Interface to Network Security Functions (I2NSF) [[I-D.ietf-i2nsf-capability](#)].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 18, 2019.

Internet-Draft Consumer-Facing Interface Information Model

July 2018

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | | |
|------------------------|--|--------------------|
| 1. | Introduction | 3 |
| 2. | Conventions used in the Document | 5 |
| 3. | Information Model for Policy | 5 |
| 3.1. | Event Sub-Model | 7 |
| 3.1.1. | Event-Map-Group | 8 |
| 3.2. | Condition Sub-Model | 8 |
| 3.3. | Action Sub-Model | 10 |
| 4. | Information Model for Multi-Tenancy | 10 |
| 4.1. | Policy-Domain | 11 |
| 4.2. | Policy-Tenant | 12 |
| 4.3. | Policy-Role | 12 |
| 4.4. | Policy-User | 12 |
| 4.5. | Policy Management Authentication Method | 13 |
| 5. | Information Model for Policy Endpoint Groups | 14 |
| 5.1. | Tag-Source | 14 |
| 5.2. | User-Group | 15 |
| 5.3. | Device-Group | 15 |
| 5.4. | Application-Group | 16 |
| 5.5. | Location-Group | 16 |
| 6. | Information Model for Threat Prevention | 17 |
| 6.1. | Threat-Feed | 17 |
| 6.2. | Custom-List | 18 |
| 6.3. | Malware-Scan-Group | 18 |
| 7. | Information Model for Telemetry Data | 18 |
| 7.1. | Telemetry-Data | 19 |
| 7.2. | Telemetry-Source | 19 |

| | |
|---|--------------------|
| 7.3. Telemetry-Destination | 20 |
| 8. Role-Based Access Control (RBAC) | 21 |
| 9. Security Considerations | 21 |
| 10. IANA Considerations | 22 |
| 11. Acknowledgments | 22 |

| | |
|--|--------------------|
| 12. Contributors | 22 |
| 13. Informative References | 22 |
| Appendix A. Changes from draft-kumar-i2nsf-client-facing-interface-im-06 | 23 |
| Authors' Addresses | 23 |

[1. Introduction](#)

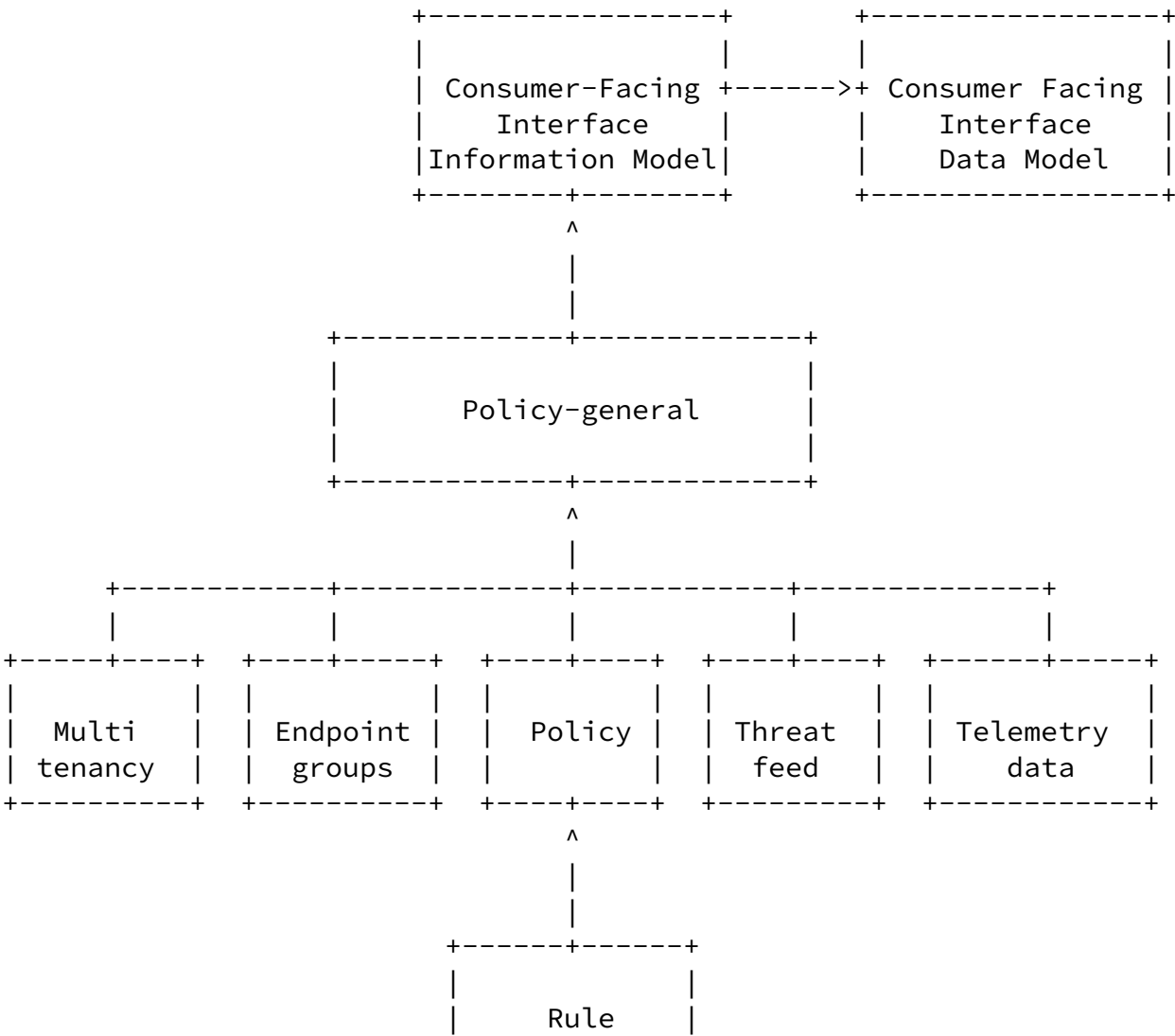
Interface to Network Security Functions (I2NSF) defines a Consumer-Facing Interface to deliver high-level security policies to Security Controller [[RFC8192](#)][RFC8329] for security enforcement in Network Security Functions (NSFs).

The Consumer-Facing Interface would be built using a set of objects, with each object capturing a unique set of information from Security Admin (i.e., I2NSF User [[RFC8329](#)]) needed to express a Security Policy. An object may have relationship with various other objects to express a complete set of requirement. An information model captures the managed objects and relationship among these objects. The information model proposed in this document is in accordance with interface requirements as defined in [[I-D.ietf-i2nsf-client-facing-interface-req](#)].

An NSF Capability model is proposed in [[I-D.ietf-i2nsf-capability](#)] as the basic model for both the NSF-Facing interface and Consumer-Facing Interface security policy model of this document. The information model proposed in this document is structured in accordance with the "Event-Condition-Event" (ECA) policy model.

[RFC3444] explains differences between an information and data model. This document use the guidelines in [[RFC3444](#)] to define an information model for Consumer-Facing Interface in this document. Figure 1 shows a high-level abstraction of Consumer-Facing Interface. A data model, which represents an implementation of the proposed information model in a specific data representation language, will be defined in a separate document.

Internet-Draft Consumer-Facing Interface Information Model July 2018



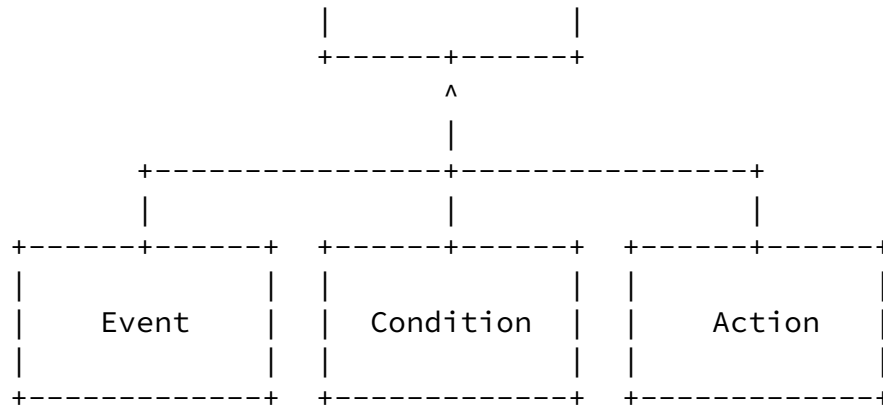


Figure 1: Diagram for High-level Abstraction of Consumer-Facing Interface

[2.](#) Conventions used in the Document

BSS: Business Support System

CLI: Command Line Interface

CMDB: Configuration Management Database

Controller: Security Controller or Management System

CRUD: Create, Retrieve, Update, Delete

FW: Firewall

GUI: Graphical User Interface

IDS: Intrusion Detection System

IPS: Intrusion Prevention System

LDAP: Lightweight Directory Access Protocol

NSF: Network Security Function, defined by
[\[I-D.ietf-i2nsf-terminology\]](#)

OSS: Operations Support System

RBAC: Role-Based Access Control

SIEM: Security Information and Event Management

URL: Universal Resource Locator

vNSF: NSF being instantiated on Virtual Machines

3. Information Model for Policy

A Policy object represents a mechanism to express a Security Policy by Security Admin (i.e., I2NSF User) using Consumer-Facing Interface toward Security Controller; the policy would be enforced on an NSF. The Policy object SHALL have following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Multi-Tenancy: The multi-tenant environment information in which the policy is applied. The Rules in the Policy can refer

to sub-objects (e.g., domain, tenant, role, and user) of it. It can be either a reference to a Multi-Tenancy object defined in another place, or a concrete object. See details in [Section 4](#).

End-Group: This field contains a list of logical entities in the business environment where a Security Policy is to be applied. It can be referenced by the Condition objects in a Rule, e.g., Source, Destination, Match, etc. It can be either a reference to an End-Group object defined in other place, or a concrete object. See details in [Section 5](#).

Threat-Feed: This field represents threat feed such as Botnet servers, GeoIP, and Malware signature. This information can be referenced by the Rule Action object directly to

execute the threat mitigation. See details in [Section 6](#).

Telemetry-Data: This field represents the telemetry collection related information that the Rule Action object can refer to about how to collect the interested telemetry information, for example, what type of telemetry to collect, where the telemetry source is, where to send the telemetry information. See details in [Section 7](#).

Rules: This field contains a list of rules. If the rule does not have a user-defined precedence, then any conflict must be manually resolved.

Owner: This field defines the owner of this policy. Only the owner is authorized to modify the contents of the policy.

A policy is a container of Rules. In order to express a Rule, a Rule must have complete information such as where and when a policy needs to be applied. This is done by defining a set of managed objects and relationship among them. A Policy Rule may be related segmentation, threat mitigation or telemetry data collection from an NSF in the network, which will be specified as the sub-model of the policy model in the subsequent sections.

The rule object SHALL have the following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event: This field includes the information to determine whether the Rule Condition can be evaluated or not. See details in [Section 3.1](#).

Condition: This field contains all the checking conditions to apply to the objective traffic. See details in [Section 3.2](#).

Action: This field identifies the action taken when a rule is matched. There is always an implicit action to drop traffic if no rule is matched for a traffic type. See details in [Section 3.3](#).

Precedence: This field identifies the precedence assigned to this rule by Security Admin. This is helpful in conflict resolution when two or more rules match a given traffic class.

[3.1](#). Event Sub-Model

The Event Object contains information related to scheduling a Rule. The Rule could be activated based on a time calendar or security event including threat level changes.

Event object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Event-Type: This field identifies whether the event of triggering policy enforcement is "ADMIN-ENFORCED", "TIME-ENFORCED" or "EVENT-ENFORCED".

Time-Information: This field contains a time calendar such as "BEGIN-TIME" and "END-TIME" for one time enforcement or recurring time calendar for periodic enforcement.

Event-Map-Group: This field contains security events or threat map in order to determine when a policy needs to be activated. This is a reference to Event-Map-Group defined later.

[3.1.1](#). Event-Map-Group

This object represents an event map containing security events and threat levels used for dynamic policy enforcement. The Event-Map-Group object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Security-Events: This contains a list of security events used for purpose for Security Policy definition.

Threat-Map: This contains a list of threat levels used for purpose for Security Policy definition.

[3.2.](#) Condition Sub-Model

This object represents Conditions that Security Admin wants to apply the checking on the traffic in order to determine whether the set of actions in the Rule can be executed or not.

The Condition object SHALL have following information:

Source: This field identifies the source of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry-Source for telemetry collection policy.

Destination: This field identifies the destination of the traffic. This could be a reference to either Policy-Endpoint-Group, Threat-Feed or Custom-List as defined earlier. This could be a special object "ALL" that matches all traffic. This could also be Telemetry- Destination for telemetry collection policy.

Match: This field identifies the match criteria used to evaluate whether the specified action needs to be taken or not. This could be either a Policy-Endpoint-Group identifying an Application set or a set of traffic rules.

Match-Direction: This field identifies whether the match criteria is to be evaluated for both directions or only one direction of the traffic with a default of allowing the other direction for stateful match conditions. This is

optional and by default a rule should apply to both directions.

Exception: This field identifies the exception consideration when a rule is evaluated for a given communication. This could be a reference to "Policy-Endpoint-Group" object or set of traffic matching criteria.

The condition object is made of condition clauses. Each condition clause consists of three tuples; variable, operator and value.

The variable and value can be source and destination IP address, for example, and they have logical operator in between to check whether they match the condition criteria set by a security admin. For Example: If condition A AND B is true: THEN execute actions ENDIF where A denotes a destination address, and B denotes a blacklisted IP address. The operator AND is the logical AND operation.

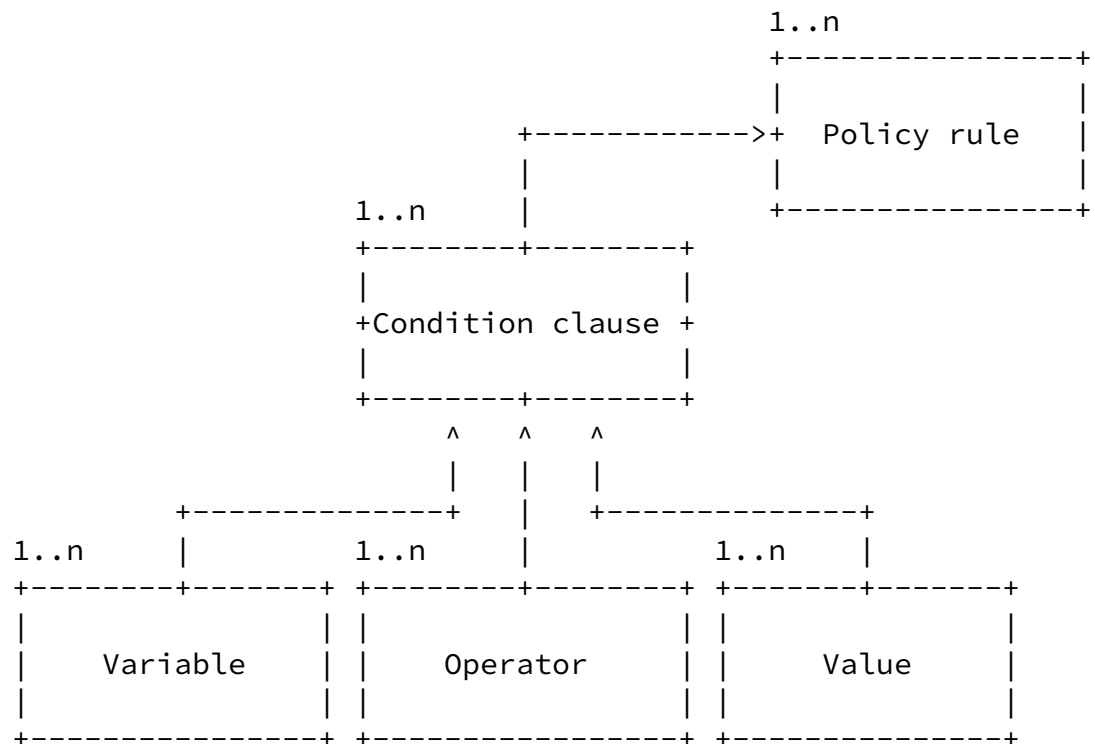


Figure 2: Condition Clause Diagram

The semantics used in a condition clause is also used in the clauses in the Event-submodel and Action sub-model.

[3.3.](#) Action Sub-Model

This object represents actions that Security Admin wants to perform based on certain traffic class.

The Action object SHALL have following information:

Name: This field identifies the name of this object.

Date: This field indicates the date when this object was created or last modified.

Primary-Action: This field identifies the action when a rule is matched by an NSF. The action could be one of "PERMIT", "DENY", "DROP-CONNECTION", "AUTHENTICATE-CONNECTION", "MIRROR", "REDIRECT", "NETFLOW", "COUNT", "ENCRYPT", "DECRYPT", "THROTTLE", "MARK", or "INSTANTIATE-NSF".

Secondary-Action: Security Admin can also specify additional actions if a rule is matched. This could be one of "LOG", "SYSLOG", or "SESSION-LOG".

[4.](#) Information Model for Multi-Tenancy

Multi-tenancy is an important aspect of any application that enables multiple administrative domains in order to manage application resources. An Enterprise organization may have multiple tenants or departments such as Human Resources (HR), Finance, and Legal, with each tenant having a need to manage their own Security Policies. In a Service Provider, a tenant could represent a Customer that wants to manage its own Security Policies.

There are multiple managed objects that constitute multi-tenancy aspects. This section lists these objects and any relationship among these objects. Below diagram shows an example of multi-tenancy in an Enterprise domain.

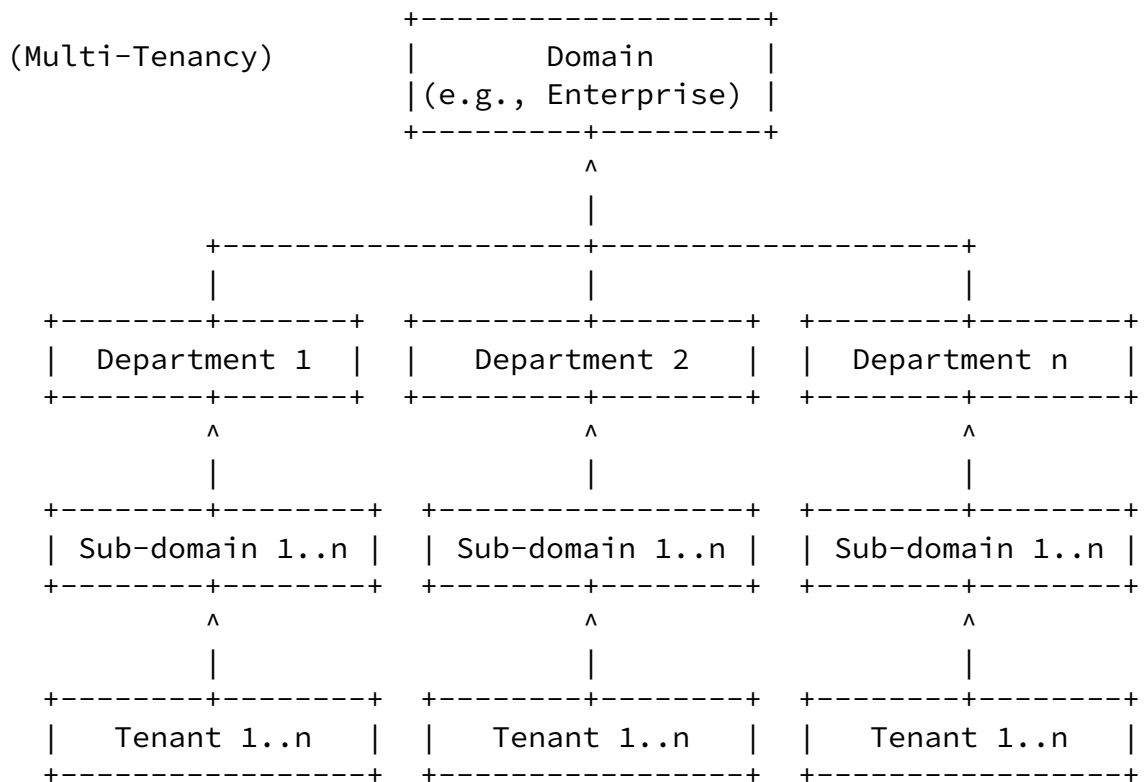


Figure 3: Multi-tenancy Diagram

[4.1.](#) Policy-Domain

This object defines a boundary for the purpose of policy management within a Security Controller. This may vary based on how the Security Controller is deployed and hosted. For example, if an Enterprise hosts a Security Controller in their network; the domain in this case could just be the one that represents that Enterprise.

But if a Cloud Service Provider hosts managed services, then a domain could represent a single customer of that Provider. Multi-tenancy model should be able to work in all such environments.

The Policy-Domain object SHALL have following information:

Name: Name of the organization or customer representing this domain.

Address: Address of the organization or customer.

Contact: Contact information of the organization or customer.

Date: Date when this account was created or last modified.

Authentication-Method: Authentication method to be used for this domain. It should be a reference to a 'Policy-Management-Authentication-Method' object.

[4.2.](#) Policy-Tenant

This object defines an entity within an organization. The entity could be a department or business unit within an Enterprise organization that would like to manage its own Policies due to regulatory compliance or business reasons.

The Policy-Tenant object SHALL have following information:

Name: Name of the Department or Division within an organization.

Date: Date when this account was created or last modified.

Domain: This field identifies the domain to which this tenant belongs. This should be a reference to a Policy-Domain object.

[4.3.](#) Policy-Role

This object defines a set of permissions assigned to a user in an organization that wants to manage its own Security Policies. It

provides a convenient way to assign policy users to a job function or a set of permissions within the organization.

The Policy-Role object SHALL have the following information:

Name: This field identifies the name of the role.

Date: Date when this role was created or last modified.

Access-Profile: This field identifies the access profile for the role. The profile grants or denies the permissions to access Endpoint Groups for the purpose of policy management or may restrict certain operations related to policy managements.

[4.4.](#) Policy-User

This object represents a unique identity within an organization. The identity authenticates with Security Controller using credentials such as a password or token in order to perform policy management. A user may be an individual, system, or application requiring access to Security Controller.

The Policy-User object SHALL have the following information:

Name: Name of a user.

Date: Date when this user was created or last modified.

Password: User password for basic authentication.

Email: E-mail address of the user.

Scope-Type: This field identifies whether the user has domain-wide or tenant-wide privileges.

Scope-Reference: This field should be a reference to either a Policy-Domain or a Policy-Tenant object.

Role: This field should be a reference to a Policy-Role object that defines the specific permissions.

[4.5.](#) Policy Management Authentication Method

This object represents authentication schemes supported by Security Controller.

This Policy-Management-Authentication-Method object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Authentication-Method: This field identifies the authentication methods. It could be a password-based, token-based, certificate-based or single sign-on authentication.

Mutual-Authentication: This field indicates whether mutual authentication is mandatory or not.

Token-Server: This field stores the information about server that validates the token submitted as credentials.

Certificate-Server: This field stores the information about server that validates certificates submitted as credentials.

Single Sign-on-Server: This field stores the information about server that validates user credentials.

[5.](#) Information Model for Policy Endpoint Groups

The Policy Endpoint Group is a very important part of building User-construct based policies. Security Admin would create and use these objects to represent a logical entity in their business environment, where a Security Policy is to be applied.

There are multiple managed objects that constitute a Policy Endpoint Group. This section lists these objects and relationship among these objects.

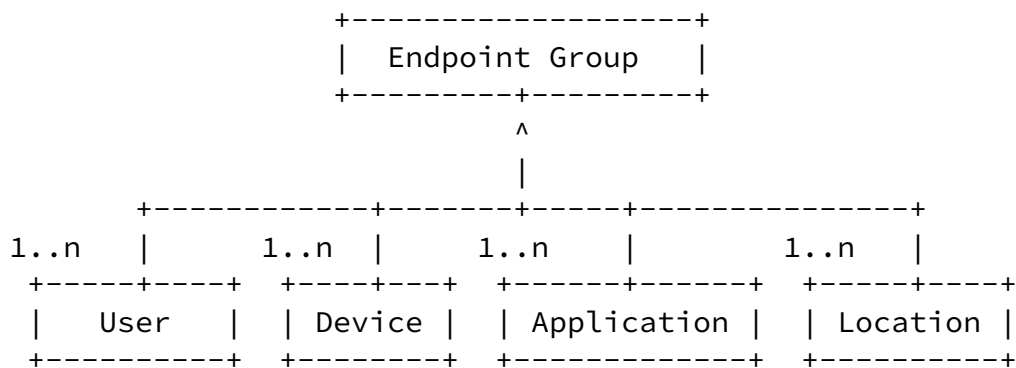


Figure 4: Endpoint Group Diagram

5.1. Tag-Source

This object represents information source for tag. The tag in a group must be mapped to its corresponding contents to enforce a Security Policy.

Tag-Source object SHALL have the following information:

Name: This field identifies name of this object.

Date: Date when this object was created or last modified.

Tag-Type: This field identifies the Endpoint Group type. It can be a User-Group, App-Group, Device-Group or Location-Group.

Tag-Source-Server: This field identifies information related to the source of the tag such as IP address and UDP/TCP port information.

Tag-Source-Application: This field identifies the protocol, e.g., LDAP, Active Directory, or CMDB used to communicate with a server.

Tag-Source-Credentials: This field identifies the credential information needed to access the server.

5.2. User-Group

This object represents a user group based on either tag or other information.

The User-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the user group is based on User-tag, User-name or IP-address.

Metadata-Server: This field should be a reference to a Metadata-Source object.

Group-Member: This field is a list of User-tag, User-names or IP addresses based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

[5.3.](#) Device-Group

This object represents a device group based on either tag or other information.

The Device-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the device group is based on Device-tag or Device-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Device-tag, Device-name or IP address based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

[5.4.](#) Application-Group

This object represents an application group based on either tag or other information.

The Application-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the application group is based on App-tag or App-name, or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Application-tag Application-name or IP address and port information based on Group-Type.

Risk-Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

[5.5.](#) Location-Group

This object represents a location group based on either tag or other information.

The 'Location-Group' object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Group-Type: This field identifies whether the location group is based on Location-tag, Location-name or IP address.

Tag-Server: This field should be a reference to a Tag-Source object.

Group-Member: This field is a list of Location-tag, Location-name

or IP addresses based on Group-Type.

Risk Level: This field represents the risk level or importance of the Endpoint to Security Admin for policy purpose; the valid range may be 0 to 9.

6. Information Model for Threat Prevention

The threat prevention plays an important part in the overall security posture by reducing the attack surfaces. This information could come in the form of threat feeds such as Botnet and GeoIP feeds usually from a third party or external service.

There are multiple managed objects that constitute this category. This section lists these objects and relationship among these objects.

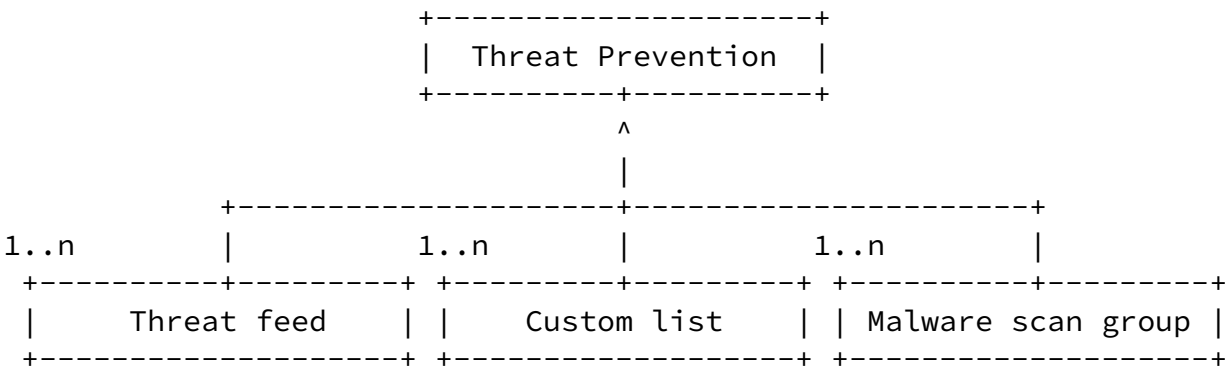


Figure 5: Threat Prevention Diagram

6.1. Threat-Feed

This object represents a threat feed such as Botnet servers and GeoIP.

The Threat-Feed object SHALL have the following information:

- Name:** This field identifies the name of this object.
- Date:** Date when this object was created or last modified.
- Feed-Type:** This field identifies whether a feed type is IP

address-based or URL-based.

Feed-Server: This field identifies the information about the feed provider, it may be an external service or local server.

Feed-Priority: This field represents the feed priority level to resolve conflict if there are multiple feed sources; the valid range may be 0 to 9.

[6.2.](#) Custom-List

This object represents a custom list created for the purpose of defining exception to threat feeds. An organization may want to allow a certain exception to threat feeds obtained from a third party

The Custom-List object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

List-Type: This field identifies whether the list type is IP address-based or URL-based.

List-Property: This field identifies the attributes of the list property, e.g., Blacklist or Whitelist.

List-Content: This field contains contents such as IP addresses or URL names.

[6.3.](#) Malware-Scan-Group

This object represents information needed to detect malware. This information could come from a local server or uploaded periodically from a third party.

The Malware-Scan-Group object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Signature-Server: This field contains information about the server from where signatures can be downloaded periodically as updates become available.

File-Types: This field contains a list of file types needed to be scanned for the virus.

Malware-Signatures: This field contains a list of malware signatures or hash values.

[7.](#) Information Model for Telemetry Data

Telemetry provides System Admin with the visibility of the network activities which can be tapped for further security analytics, e.g., detecting potential vulnerabilities, malicious activities, etc.

Kumar, et al. Expires January 18, 2019 [Page 18]

Internet-Draft Consumer-Facing Interface Information Model July 2018

[7.1.](#) Telemetry-Data

This object contains information collected for telemetry.

The Telemetry-Data object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Log-Data: This field identifies whether Log data need to be collected.

Syslog-Data This field identifies whether Syslog data need to be collected.

SNMP-Data: This field identifies whether SNMP traps and alarm data need to be collected.

sFlow-Record: This field identifies whether sFlow records need to be collected.

NetFlow-Record: This field identifies whether NetFlow record need to be collected.

NSF-Stats: This field identifies whether statistics need to be

collected from an NSF.

[7.2.](#) Telemetry-Source

This object contains information related to telemetry source. The source would be an NSF in the network.

The Telemetry-Source object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Source-Type: This field contains the type of the NSF telemetry source: "NETWORK-NSF", "FIREWALL-NSF", "IDS-NSF", "IPS-NSF", "PROXY-NSF" or "OTHER-NSF".

NSF-Source: This field contains information such as IP address and protocol (UDP or TCP) port number of the NSF providing telemetry data.

NSF-Credentials: This field contains a username and a password used to authenticate the NSF.

Collection-Interval: This field contains time in milliseconds between each data collection. For example, a value of 5,000 means data is streamed to collector every 5 seconds. Value of 0 means data streaming is event-based.

Collection-Method: This field contains a method of collection whether it is PUSH-based or PULL-based.

Heartbeat-Interval: This field contains time in seconds when the source must send telemetry heartbeat.

QoS-Marking: This field contains a DSCP value source marked on its generated telemetry packets.

[7.3.](#) Telemetry-Destination

This object contains information related to telemetry destination. The destination is usually a collector which is either a part of Security Controller or external system such as SIEM.

The Telemetry-Destination object SHALL have the following information:

Name: This field identifies the name of this object.

Date: Date when this object was created or last modified.

Collector-Source: This field contains the information such as IP address and protocol (UDP or TCP) port number for the collector's destination.

Collector-Credentials: This field contains a username and a password provided by the collector.

Data-Encoding: This field contains the telemetry data encoding, which could in the form of a schema.

Data-Transport: This field contains streaming telemetry data protocols: whether it is gRPC, protocol buffer over UDP, etc.

[8.](#) Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) provides a powerful and centralized control within a network. It is a policy neutral access control mechanism defined around roles and privileges. The components of RBAC, such as role-permissions, user-role and role-role relationships, make it simple to perform user assignments.

```
+-----+
|       |
|  User 1  + (has many)
|       | \
```

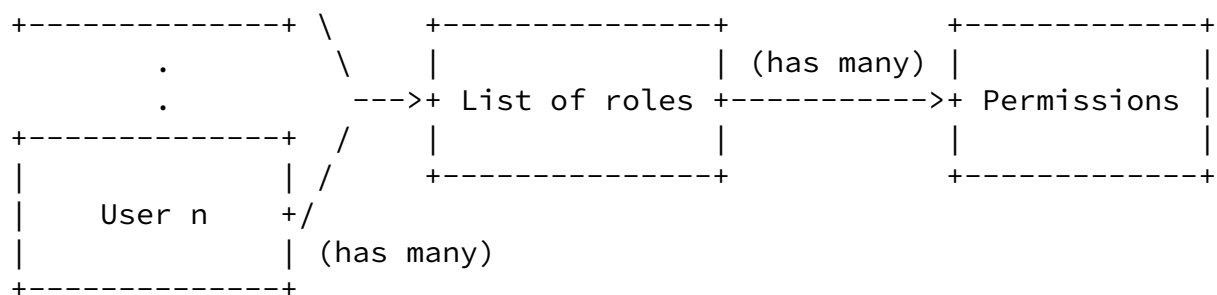


Figure 6: RBAC Diagram

As shown in Figure 6, a role represents a collection of permissions (e.g., accessing a file server or other particular resources). A role may be assigned to one or multiple users. Both roles and permissions can be organized in a hierarchy. A role may consists of other roles and permissions.

Following are the steps required to build RBAC.

1. Defining roles and permissions.
2. Establishing relations among roles and permissions.
3. Defining users.
4. Associating rules with roles and permissions.
5. assigning roles to users.

9. Security Considerations

An information model provides a mechanism to protect Consumer-Facing Interface between System Admin (i.e., I2NSF User) and Security Controller. One of the specified mechanism must be used to protect an Enterprise network, data and all resources from external attacks. This information model mandates that the interface must have proper

authentication and authorization with Role-Based Access Controls to address the multi-tenancy requirement. The document does not mandate that a particular mechanism should be used because a different organization may have different needs based on their deployment.

10. IANA Considerations

This document requires no IANA actions. RFC Editor: Please remove this section before publication.

11. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

12. Contributors

This document is the work of I2NSF working group, greatly benefiting from inputs and suggestions by Kunal Modasiya, Prakash T. Sehsadri and Srinivas Nimmagadda from Juniper Networks. The authors sincerely appreciate their contributions.

The following are contributing authors of this document, who are considered co-authors:

- o Eunsoo Kim (Sungkyunkwan University)
- o Seungjin Lee (Sungkyunkwan University)
- o Hyounghick Kim (Sungkyunkwan University)

13. Informative References

[I-D.ietf-i2nsf-capability]

Xia, L., Strassner, J., Basile, C., and D. Lopez, "Information Model of NSFs Capabilities", [draft-ietf-i2nsf-capability-02](#) (work in progress), July 2018.

[I-D.ietf-i2nsf-client-facing-interface-req]

Kumar, R., Lohiya, A., Qi, D., Bitar, N., Palislamovic, S., and L. Xia, "Requirements for Client-Facing Interface to Security Controller", [draft-ietf-i2nsf-client-facing-interface-req-05](#) (work in progress), May 2018.

[I-D.ietf-i2nsf-terminology]

Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", [draft-ietf-i2nsf-terminology-06](#) (work in progress), July 2018.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", [RFC 3444](#), DOI 10.17487/RFC3444, January 2003, <<https://www.rfc-editor.org/info/rfc3444>>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", [RFC 8192](#), DOI 10.17487/RFC8192, July 2017, <<https://www.rfc-editor.org/info/rfc8192>>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", [RFC 8329](#), DOI 10.17487/RFC8329, February 2018, <<https://www.rfc-editor.org/info/rfc8329>>.

Appendix A. Changes from [draft-kumar-i2nsf-client-facing-interface-im-06](#)

The following changes have been made from [draft-kumar-i2nsf-client-facing-interface-im-06](#):

- o In [Section 1](#), Figure 1 is added to show a diagram for a high-level abstraction of Consumer-Facing Interface.

Authors' Addresses

Rakesh Kumar
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: rakeshkumarccloud@gmail.com

Internet-Draft Consumer-Facing Interface Information Model

July 2018

Anil Lohiya
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
US

Email: alohiya@juniper.net

Dave Qi
Bloomberg
731 Lexington Avenue
New York, NY 10022
US

Email: DQI@bloomberg.net

Nabil Bitar
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: nabil.bitar@nokia.com

Senad Palislamovic
Nokia
755 Ravendale Drive
Mountain View, CA 94043
US

Email: senad.palislamovic@nokia.com

Liang Xia
Huawei
101 Software Avenue
Nanjing, Jiangsu 210012
China

Email: Frank.Xialiang@huawei.com

Kumar, et al.

Expires January 18, 2019

[Page 24]

Internet-Draft Consumer-Facing Interface Information Model

July 2018

Jaehoon Paul Jeong
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4957

Fax: +82 31 290 7996

Email: pauljeong@skku.edu

URI: <http://iotlab.skku.edu/people-jaehoon-jeong.php>

