Workgroup: I2NSF Working Group
Internet-Draft:
draft-yang-i2nsf-security-policy-
translation-12
Published: 24 October 2022
Intended Status: Standards Track
Expires: 27 April 2023
Authors: J. Jeong, Ed.
Sungkyunkwan University
P. Lingga
Sungkyunkwan University
J. Yang
Sungkyunkwan University
J. Kim
Sungkyunkwan University

Guidelines for Security Policy Translation in Interface to Network Security Functions

Abstract

This document proposes the guidelines for security policy translation in Interface to Network Security Functions (I2NSF) Framework. When I2NSF User delivers a high-level security policy for a security service, Security Policy Translator in Security Controller translates it into a low-level security policy for Network Security Functions (NSFs). For this security policy translation, this document specifies the relation between a high-level security policy based on the Consumer-Facing Interface YANG data model and a low-level security policy based on the NSF-Facing Interface YANG data model. Also, it describes an architecture of a security policy translator along with an NSF database, and the process of security policy translation with the NSF database.

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 27 April 2023.

Table of Contents

1. Introduction

This document proposes the guidelines for security policy translation in Interface to Network Security Functions (I2NSF) Framework [RFC8329]. First of all, this document explains the necessity of a security policy translator (shortly called policy translator) in the I2NSF framework.

The policy translator resides in Security Controller in the I2NSF framework and translates a high-level security policy to a low-level security policy for Network Security Functions (NSFs). A high-level policy is specified by I2NSF User in the I2NSF framework and is delivered to Security Controller via Consumer-Facing Interface [I-D.ietf-i2nsf-consumer-facing-interface-dm]. It is translated into a low-level policy by Policy Translator in Security Controller and is delivered to NSFs to execute the rules corresponding to the low-level policy via NSF-Facing Interface [I-D.ietf-i2nsf-nsf-facing-interface-dm].

2. Terminology

This document uses the terminology specified in [RFC8329].

3. Necessity for Security Policy Translator

Security Controller acts as a coordinator between I2NSF User and NSFs. Also, Security Controller has capability information of NSFs that are registered via Registration Interface [I-D.ietf-i2nsf-registration-interface-dm] by Developer's Management System [RFC8329]. As a coordinator, Security Controller needs to generate a low-level policy in the form of security rules intended by the high-level policy, which can be understood by the corresponding NSFs.

The high-level and low-level security policies are specified by YANG data model [RFC7950] with the delivery using either NETCONF [RFC6241] or RESTCONF [RFC8040]. The translation from a high-level security policy to the corresponding low-level security policy will be able to rapidly elevate I2NSF in real-world deployment. A rule in a high-level policy can include a broad target object, such as employees in a company for a security service (e.g., firewall and web filter). Such employees may be from human resource (HR) department, software engineering department, and advertisement

department. A keyword of employee needs to be mapped to these employees from various departments. This mapping needs to be handled by a security policy translator in a flexible way while understanding the intention of a policy specification. Let us consider the following two policies:

*Block my son's computers from malicious websites.

*Drop packets from the IP address 192.0.2.0/24 to malicious1 and malicious2.

The above two sentences are examples of policies for blocking malicious websites. Both policies are for the same operation. However, NSF cannot understand the first policy, because the policy does not have any specified information for NSF. To set up the policy at an NSF, the NSF MUST receive at least the source IP address and website address for an operation. It means that the first sentence is NOT compatible for an NSF policy. Conversely, when I2NSF Users request a security policy to the system, they never make a security policy like the second example. For generating a security policy like the second sentence, the user MUST know that the NSF needs to receive the specified information, source IP address and website address. It means that the user understands the NSF professionally, but there are not many professional users in a small size of company or at a residential area. In conclusion, the I2NSF User prefers to issue a security policy in the first sentence, but an NSF will require the same policy as the second sentence with specific information. Therefore, an advanced translation scheme of security policy is REQUIRED in I2NSF.

This document proposes an approach using Automata theory [Automata] for the policy translation, such as Deterministic Finite Automaton (DFA). Note that Automata theory is the foundation of programming language and compiler. Thus, with this approach, I2NSF User can easily specify a high-level security policy that will be enforced into the corresponding NSFs with a compatibly low-level security policy with the help of Security Policy Translator. Also, for easy management, a modularized translator structure is proposed.

4. Relation between Consumer-Facing Interface and NSF-Facing Interface YANG Data Models

The Consumer-Facing Interface (CFI) YANG data model [I-D.ietf-i2nsf-consumer-facing-interface-dm] and NSF-Facing Interface (NFI) YANG data model [I-D.ietf-i2nsf-nsf-facing-interface-dm] are two data models designed with different objectives in mind. The CFI is designed to be used by someone with little knowledge of network security can configure the NSFs by specifying the required information, their

data types, and encoding schemes as a high-level policy. The NFI is designed to provide detailed security policy configuration for the NSFs as a low-level policy that can be used by the NSFs to deploy security services. But even with the distinct objectives for the data models, the attributes between the two data models are constructed to have a relation for the purpose of automation. Thus, this section provides the information of the relationship between the attributes in the CFI and NFI YANG data model.

4.1. The CFI and NFI Top-Level YANG Trees Comparison

```
Consumer-Facing Interface (CFI):
module: ietf-i2nsf-cfi-policy
  +--rw i2nsf-cfi-policy* [name]
     +--rw name                   string
     +--rw language?              string
     +--rw resolution-strategy?   identityref
     +--rw rules* [name]
     |  ...
     +--rw endpoint-groups
     |  ...
     +--rw threat-prevention
        ...

NSF-Facing Interface (NFI):
module: ietf-i2nsf-policy-rule-for-nsf
  +--rw i2nsf-security-policy* [name]
     +--rw name                   string
     +--rw language?              string
     +--rw priority-usage?        identityref
     +--rw resolution-strategy?   identityref
     +--rw default-action?        identityref
     +--rw rules* [name]
     |  ...
     +--rw rule-group
        ...
```

Figure 1: The CFI and NFI Top-Level YANG Trees

Figure 1 shows the top-level of the CFI and NFI YANG Trees. The CFI and NFI top-level provides the basic security policy information such as name of a policy, language tag, and resolution-strategy. Both data models also provide list of rules to be executed to perform the network security services.

The differences of the top-level data models are default action and priority usage are not provided in CFI YANG data model. This is because the philosophy of CFI, i.e., To make CFI as simple as

possible for the user. But this attributes can be given by the Security Controller with a default value in the translation process. Another important distinct point is CFI YANG data model also provides endpoint groups and threat prevention to register high-level information (e.g., mapping a user to an IP address) to the database for high-level configuration that can be used to translate the high-level policy into the low-level policy.

4.2. The CFI and NFI Rule-Level YANG Trees Comparison

```
Consumer-Facing Interface (CFI):
 +--rw rules* [name]
     |  +--rw name         string
     |  +--rw priority?    uint8
     |  +--rw event
     |  |  ...
     |  +--rw condition
     |  |  ...
     |  +--rw action
     |     ...

NSF-Facing Interface (NFI):
 +--rw rules* [name]
     |  +--rw name               string
     |  +--rw description?       string
     |  +--rw priority?          uint8
     |  +--rw enable?            boolean
     |  +--rw long-connection
     |  |  +--rw enable?     boolean
     |  |  +--rw duration?   uint32
     |  +--rw event
     |  |  ...
     |  +--rw condition
     |  |  ...
     |  +--rw action
     |     ...
```

Figure 2: The CFI and NFI Rule-Level YANG Trees

Figure 2 shows the rule-level YANG trees of the CFI and NFI YANG Trees. Similarly to the top-level YANG data model, the long-connection is not provided in the CFI YANG data model to simplify the data model for the user configuration. This value can also be added using a default value in the Security Controller for the low-level security policy.

In term of similarity, the CFI and NFI YANG data model provides the basic rule information such as the unique name the priority value

for the rules. Both data models utilize the Event-Condition-Action (ECA) policy rule described in Section 3.1 of the [I-D.ietf-i2nsf-capability-data-model].

4.2.1. The CFI and NFI Event YANG Data Models Comparison

```
Consumer-Facing Interface (CFI):
|  +--rw event
|  |  +--rw system-event*   identityref
|  |  +--rw system-alarm*   identityref

NSF-Facing Interface (NFI):
|  +--rw event
|  |  +--rw description?     string
|  |  +--rw system-event*   identityref
|  |  +--rw system-alarm*   identityref
```

Figure 3: The CFI and NFI Event YANG Trees

As shown in Figure 3, CFI and NFI YANG data models have the almost same structures for Event except for description in NFI. The description is optional because it contains human-readable text for the description of an event.

4.2.2. The CFI and NFI Condition YANG Data Models Comparison

```
Consumer-Facing Interface (CFI):
|  +--rw condition
|  |  +--rw firewall
|  |  |  +--rw source*                     union
|  |  |  +--rw destination*                union
|  |  |  +--rw transport-layer-protocol?   identityref
|  |  |  +--rw range-port-number
|  |  |  |  +--rw start-port-number?   inet:port-number
|  |  |  |  +--rw end-port-number?     inet:port-number
|  |  |  +--rw icmp
|  |  |     +--rw message*   identityref
|  |  +--rw ddos
|  |  |  +--rw rate-limit
|  |  |     +--rw packet-rate-threshold?   uint64
|  |  |     +--rw byte-rate-threshold?     uint64
|  |  |     +--rw flow-rate-threshold?     uint64
|  |  +--rw anti-virus
|  |  |  +--rw exception-files*   string
|  |  +--rw payload
|  |  |  +--rw content*
          -> /i2nsf-cfi-policy/threat-prevention/payload-content/name
|  |  +--rw url-category
|  |  |  +--rw url-name?
                  -> /i2nsf-cfi-policy/endpoint-groups/url-group/name
|  |  +--rw voice
|  |  |  +--rw source-id*        string
|  |  |  +--rw destination-id*   string
|  |  |  +--rw user-agent*       string
|  |  +--rw context
|  |  |  +--rw time
|  |  |  |  +--rw start-date-time?   yang:date-and-time
|  |  |  |  +--rw end-date-time?     yang:date-and-time
|  |  |  |  +--rw period
|  |  |  |  |  +--rw start-time?   time
|  |  |  |  |  +--rw end-time?     time
|  |  |  |  |  +--rw day*          day
|  |  |  |  |  +--rw date*         int32
|  |  |  |  |  +--rw month*        string
|  |  |  |  +--rw frequency?         enumeration
|  |  |  +--rw application
|  |  |  |  +--rw protocol*      identityref
|  |  |  +--rw device-type
|  |  |  |  +--rw device*        identityref
|  |  |  +--rw users
|  |  |  |  +--rw user* [id]
|  |  |  |  |  +--rw id      uint32
|  |  |  |  |  +--rw name?   string
|  |  |  |  +--rw group* [id]
|  |  |  |     +--rw id      uint32
```

```
|  |  |  |     +--rw name?   string
|  |  |  +--rw geographic-location
|  |  |     +--rw source*
|  |  |      -> /i2nsf-cfi-policy/endpoint-groups/location-group/name
|  |  |     +--rw destination*
|  |  |      -> /i2nsf-cfi-policy/endpoint-groups/location-group/name
|  |  +--rw threat-feed
|  |     +--rw name*
|  |      -> /i2nsf-cfi-policy/threat-prevention/threat-feed-list/name

NSF-Facing Interface:
     |  +--rw condition
     |  |  +--rw description?    string
     |  |  +--rw layer-2* [destination-mac-address source-mac-address
     |  |                  ethertype]
     |  |  |  +--rw description?                    string
     |  |  |  +--rw destination-mac-address         yang:mac-address
     |  |  |  +--rw destination-mac-address-mask?   yang:mac-address
     |  |  |  +--rw source-mac-address              yang:mac-address
     |  |  |  +--rw source-mac-address-mask?        yang:mac-address
     |  |  |  +--rw ethertype                       eth:ethertype
     |  |  +--rw (layer-3)?
     |  |  |  +--:(ipv4)
     |  |  |  |  +--rw ipv4
     |  |  |  |     +--rw description?                    string
     |  |  |  |     +--rw dscp?                           inet:dscp
     |  |  |  |     +--rw ecn?                            uint8
     |  |  |  |     +--rw length?                         uint16
     |  |  |  |     +--rw ttl?                            uint8
     |  |  |  |     +--rw protocol?                       uint8
     |  |  |  |     +--rw ihl?                            uint8
     |  |  |  |     +--rw flags?                          bits
     |  |  |  |     +--rw offset?                         uint16
     |  |  |  |     +--rw identification?                 uint16
     |  |  |  |     +--rw (destination-network)?
     |  |  |  |     |  +--:(destination-ipv4-network)
     |  |  |  |     |  |  +--rw destination-ipv4-network?
                                                  inet:ipv4-prefix
     |  |  |  |     |  +--:(destination-ipv4-range)
     |  |  |  |     |     +--rw destination-ipv4-range* [start end]
     |  |  |  |     |        +--rw start    inet:ipv4-address-no-zone
     |  |  |  |     |        +--rw end      inet:ipv4-address-no-zone
     |  |  |  |     +--rw (source-network)?
     |  |  |  |        +--:(source-ipv4-network)
     |  |  |  |        |  +--rw source-ipv4-network? inet:ipv4-prefix
     |  |  |  |        +--:(source-ipv4-range)
     |  |  |  |           +--rw source-ipv4-range* [start end]
     |  |  |  |              +--rw start    inet:ipv4-address-no-zone
     |  |  |  |              +--rw end      inet:ipv4-address-no-zone
```

```
|  |  |  +--:(ipv6)
|  |  |     +--rw ipv6
|  |  |        +--rw description?                    string
|  |  |        +--rw dscp?                           inet:dscp
|  |  |        +--rw ecn?                            uint8
|  |  |        +--rw length?                         uint16
|  |  |        +--rw ttl?                            uint8
|  |  |        +--rw protocol?                       uint8
|  |  |        +--rw (destination-network)?
|  |  |        |  +--:(destination-ipv6-network)
|  |  |        |  |  +--rw destination-ipv6-network?
                                              inet:ipv6-prefix
|  |  |        |  +--:(destination-ipv6-range)
|  |  |        |     +--rw destination-ipv6-range* [start end]
|  |  |        |        +--rw start    inet:ipv6-address-no-zone
|  |  |        |        +--rw end      inet:ipv6-address-no-zone
|  |  |        +--rw (source-network)?
|  |  |        |  +--:(source-ipv6-network)
|  |  |        |  |  +--rw source-ipv6-network? inet:ipv6-prefix
|  |  |        |  +--:(source-ipv6-range)
|  |  |        |     +--rw source-ipv6-range* [start end]
|  |  |        |        +--rw start    inet:ipv6-address-no-zone
|  |  |        |        +--rw end      inet:ipv6-address-no-zone
|  |  |        +--rw flow-label?      inet:ipv6-flow-label
|  |  +--rw (layer-4)?
|  |  |  +--:(tcp)
|  |  |  |  +--rw tcp
|  |  |  |     +--rw description?              string
|  |  |  |     +--rw source-port-number
|  |  |  |     |  +--rw (source-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw destination-port-number
|  |  |  |     |  +--rw (destination-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
```

```
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw sequence-number?           uint32
|  |  |  |     +--rw acknowledgement-number?    uint32
|  |  |  |     +--rw data-offset?               uint8
|  |  |  |     +--rw reserved?                  uint8
|  |  |  |     +--rw flags?                     bits
|  |  |  |     +--rw window-size?               uint16
|  |  |  |     +--rw urgent-pointer?            uint16
|  |  |  |     +--rw options?                   binary
|  |  |  +--:(udp)
|  |  |  |  +--rw udp
|  |  |  |     +--rw description?               string
|  |  |  |     +--rw source-port-number
|  |  |  |     |  +--rw (source-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw destination-port-number
|  |  |  |     |  +--rw (destination-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw length?                    uint16
|  |  |  +--:(sctp)
|  |  |  |  +--rw sctp
```

```
|  |  |  |     +--rw description?              string
|  |  |  |     +--rw source-port-number
|  |  |  |     |  +--rw (source-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw destination-port-number
|  |  |  |     |  +--rw (destination-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw chunk-type*               uint8
|  |  |  |     +--rw chunk-length?             uint16
|  |  |  +--:(dccp)
|  |  |  |  +--rw dccp
|  |  |  |     +--rw description?              string
|  |  |  |     +--rw source-port-number
|  |  |  |     |  +--rw (source-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw destination-port-number
```

```
|  |  |  |     |  +--rw (destination-port)?
|  |  |  |     |     +--:(range-or-operator)
|  |  |  |     |     |  +--rw (port-range-or-operator)?
|  |  |  |     |     |     +--:(range)
|  |  |  |     |     |     |  +--rw lower-port  inet:port-number
|  |  |  |     |     |     |  +--rw upper-port  inet:port-number
|  |  |  |     |     |     +--:(operator)
|  |  |  |     |     |        +--rw operator?     operator
|  |  |  |     |     |        +--rw port        inet:port-number
|  |  |  |     |     +--:(port-list)
|  |  |  |     |        +--rw port-numbers* [start end]
|  |  |  |     |           +--rw start    inet:port-number
|  |  |  |     |           +--rw end      inet:port-number
|  |  |  |     +--rw service-code*              uint32
|  |  |  |     +--rw type*                      uint8
|  |  |  |     +--rw data-offset?               uint8
|  |  |  +--:(icmp)
|  |  |     +--rw icmp
|  |  |        +--rw description?      string
|  |  |        +--rw version?          enumeration
|  |  |        +--rw type?             uint8
|  |  |        +--rw code?             uint8
|  |  |        +--rw rest-of-header?   binary
|  |  +--rw url-category
|  |  |  +--rw description?    string
|  |  |  +--rw pre-defined*    string
|  |  |  +--rw user-defined*   string
|  |  +--rw voice
|  |  |  +--rw description?            string
|  |  |  +--rw source-voice-id*        string
|  |  |  +--rw destination-voice-id*   string
|  |  |  +--rw user-agent*             string
|  |  +--rw ddos
|  |  |  +--rw description?         string
|  |  |  +--rw alert-packet-rate?   uint32
|  |  |  +--rw alert-flow-rate?     uint32
|  |  |  +--rw alert-byte-rate?     uint32
|  |  +--rw anti-virus
|  |  |  +--rw profile*           string
|  |  |  +--rw exception-files*   string
|  |  +--rw payload
|  |  |  +--rw description?   string
|  |  |  +--rw content*       binary
|  |  +--rw context
|  |     +--rw description?           string
|  |     +--rw time
|  |     |  +--rw start-date-time?   yang:date-and-time
|  |     |  +--rw end-date-time?     yang:date-and-time
|  |     |  +--rw period
```

```
|  |     |  |  +--rw start-time?   time
|  |     |  |  +--rw end-time?     time
|  |     |  |  +--rw day*          day
|  |     |  |  +--rw date*         int8
|  |     |  |  +--rw month*        string
|  |     |  +--rw frequency?         enumeration
|  |     +--rw application
|  |     |  +--rw description?   string
|  |     |  +--rw protocol*      identityref
|  |     +--rw device-type
|  |     |  +--rw description?   string
|  |     |  +--rw device*        identityref
|  |     +--rw users
|  |     |  +--rw description?   string
|  |     |  +--rw user* [id]
|  |     |  |  +--rw id      uint32
|  |     |  |  +--rw name?   string
|  |     |  +--rw group* [id]
|  |     |     +--rw id      uint32
|  |     |     +--rw name?   string
|  |     +--rw geographic-location
|  |        +--rw description?   string
|  |        +--rw source*        string
|  |        +--rw destination*   string
```

Figure 4: The CFI and NFI Condition YANG Trees

Figure 4 shows CFI and NFI Condition YANG Trees. It shows a different way to manipulate the Access Control Lists (ACLs) for the CFI and NFI YANG data models. The CFI aims at an easy security policy configuration, thus only provides a simple and most often needed fields in ACls, i.e., source and destination address (IPv4 or IPv6), type of transport protocol, source and destination port numbers, type of application protocol, and ICMP type and code. While, the NFI imports from [RFC8519] to provide a detailed configuration of packet header.

Additionally, both data models provide configuration for advanced network security functions such as DDoS, Antivirus, Payload (DPI), URL Filtering, and Voice Filtering conditions. The difference is that in CFI some of the information (name, value) for configuration is saved into a database in Security Controller for easy configuration. The configuration can be done by using the key name that holds the corresponding value.

The YANG data models also has context condition that can be one to one mapped, such as time condition to define the active period of a rule or geographic location condition to filter traffic from/to a certain region that can be mapped into the source and destination IP (IPv4 or IPv6) addresses based on the database provided.

4.2.3. The CFI and NFI Action YANG Data Models Comparison

```
Consumer-Facing Interface (CFI):
    +--rw action
    |  +--rw primary-action
    |  |  +--rw action?   identityref
    |  +--rw secondary-action
    |     +--rw log-action?   identityref

NSF-Facing Interface (NFI):
 |  +--rw action
 |     +--rw description?       string
 |     +--rw packet-action
 |     |  +--rw ingress-action?   identityref
 |     |  +--rw egress-action?    identityref
 |     |  +--rw log-action?       identityref
 |     +--rw flow-action
 |     |  +--rw ingress-action?   identityref
 |     |  +--rw egress-action?    identityref
 |     |  +--rw log-action?       identityref
 |     +--rw advanced-action
 |        +--rw content-security-control*    identityref
 |        +--rw attack-mitigation-control*   identityref
```

Figure 5: The CFI and NFI Action YANG Trees

Figure 4 shows CFI and NFI Action YANG Trees. The action in CFI YANG data model is separated into primary-action and secondary-action. Primary action is the Ingress and Egress action (i.e., pass, drop, reject, rate-limit, mirror, invoke-signaling, tunnel-encapsulation, forwarding, and transformation) in the NFI YANG data model. The secondary-action is the log-action to log the rule that has been triggered by a packet/flow or log the packet/flow that triggered the rule. The NFI also can specify the action as packet or flow action depending on the capability of the NSF.

In NFI YANG data model, the advanced action is used to activate the Service Function Chaining (SFC) to apply multiple NSFs on network traffics. This does not exist in CFI as the CFI is used to provide a high-level action. The action of a certain policy in CFI may require multiple NSFs (e.g., a URL filtering with firewall) as a single NSF may not have the capability to handle the security policy. Thus, the SFC of those NSFs is handled by NFI.

5. Design of Security Policy Translator

Commonly used security policies are created as XML (Extensible Markup Language) [XML] files. A popular way to change the format of an XML file is to use an XSLT (Extensible Stylesheet Language Transformation) [XSLT] document. XSLT is an XML-based language to transform an input XML file into another output XML file. However,

the use of XSLT makes it difficult to manage the security policy translator and to handle the registration of new capabilities of NSFs. With the necessity for a security policy translator, this document describes a security policy translator based on Automata theory.

5.1. Overall Structure of Security Policy Translator

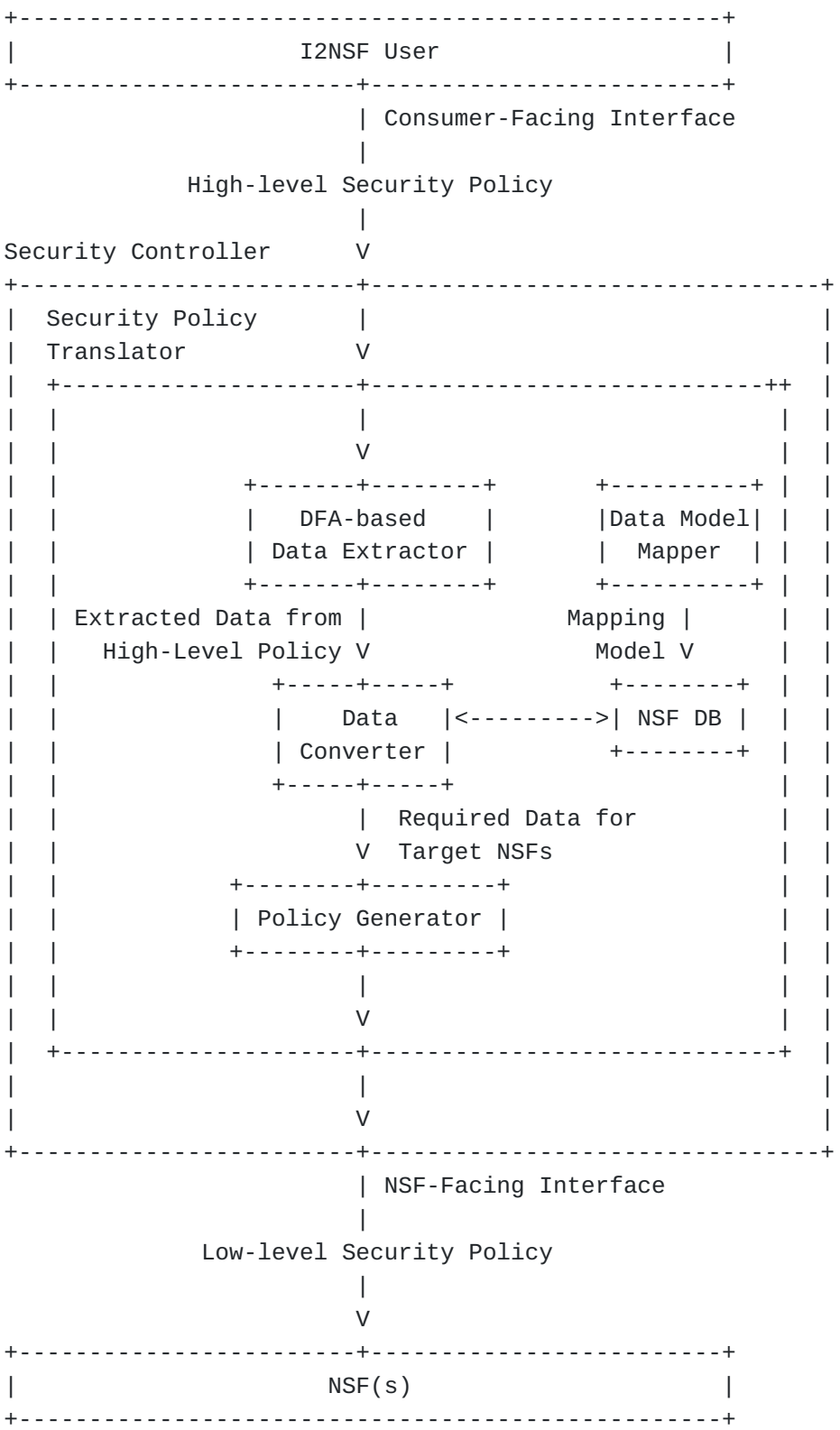


Figure 6: The Overall Design of Security Policy Translator

Figure 6 shows the overall design for Security Policy Translator in Security Controller. There are four main components for Security

Policy Translator: Data Extractor, Data Converter, Policy Generator, and Data Model Mapper.

Extractor is a DFA-based module for extracting data from a high-level policy which I2NSF User delivered via Consumer-Facing Interface. Data Model Mapper creates a mapping model for mapping the elements between Consumer-Facing Interface and NSF-Facing Interface. Data Converter converts the extracted data to the capabilities of target NSFs for a low-level policy. It refers to an NSF Database (DB) in order to convert an abstract subject or object into the corresponding concrete subject or object (e.g., IP address and website URL). Policy Generator generates a low-level policy which will execute the NSF capabilities from Converter.

5.2. DFA-based Data Extractor

5.2.1. Design of DFA-based Data Extractor

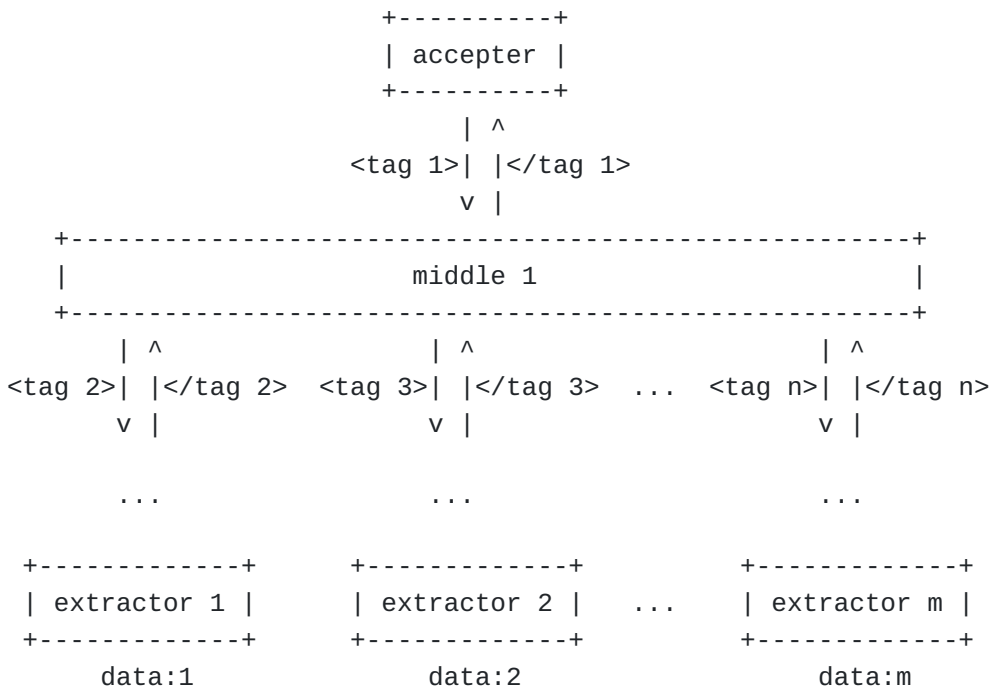


Figure 7: DFA Architecture of Data Extractor

Figure 7 shows a design for Data Extractor in the security policy translator. If a high-level policy contains data along the hierarchical structure of the standard Consumer-Facing Interface YANG data model [I-D.ietf-i2nsf-consumer-facing-interface-dm], data can be easily extracted using the state transition machine, such as DFA. The extracted data can be processed and used by an NSF to understand it. Extractor can be constructed by designing a DFA with the same hierarchical structure as a YANG data model.

After constructing a DFA, Data Extractor can extract all of data in the entered high-level policy by using state transitions. Also, the DFA can easily detect the grammar errors of the high-level policy. The extracting algorithm of Data Extractor is as follows:

1. Start from the 'accepter' state.

2. Read the next tag from the high-level policy.

3. Transit to the corresponding state.

4. If the current state is in 'extractor', extract the corresponding data, and then go back to step 2.

5. If the current state is in 'middle', go back to step 2.

6. If there is no possible transition and arrived at 'accepter' state, the policy has no grammar error. Otherwise, there is a grammar error, so stop the process with failure.

5.2.2. Example Scenario for Data Extractor

```
<i2nsf-cfi-policy
  xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-cfi-policy">
  <name>block_web_security_policy</name>
  <rules>
    <name>block_web</name>
    <condition>
      <firewall-condition>
        <source>Son's_PC</source>
      <firewall-condition>
      <url-condition>
        <url-name>malicious_websites</url-name>
      </url-condition>
    </condition>
    <actions>
      <primary-action>
        <action>drop</action>
      </primary-action>
    </actions>
  </rules>
</i2nsf-cfi-policy>
```

Figure 8: The Example of High-level Policy

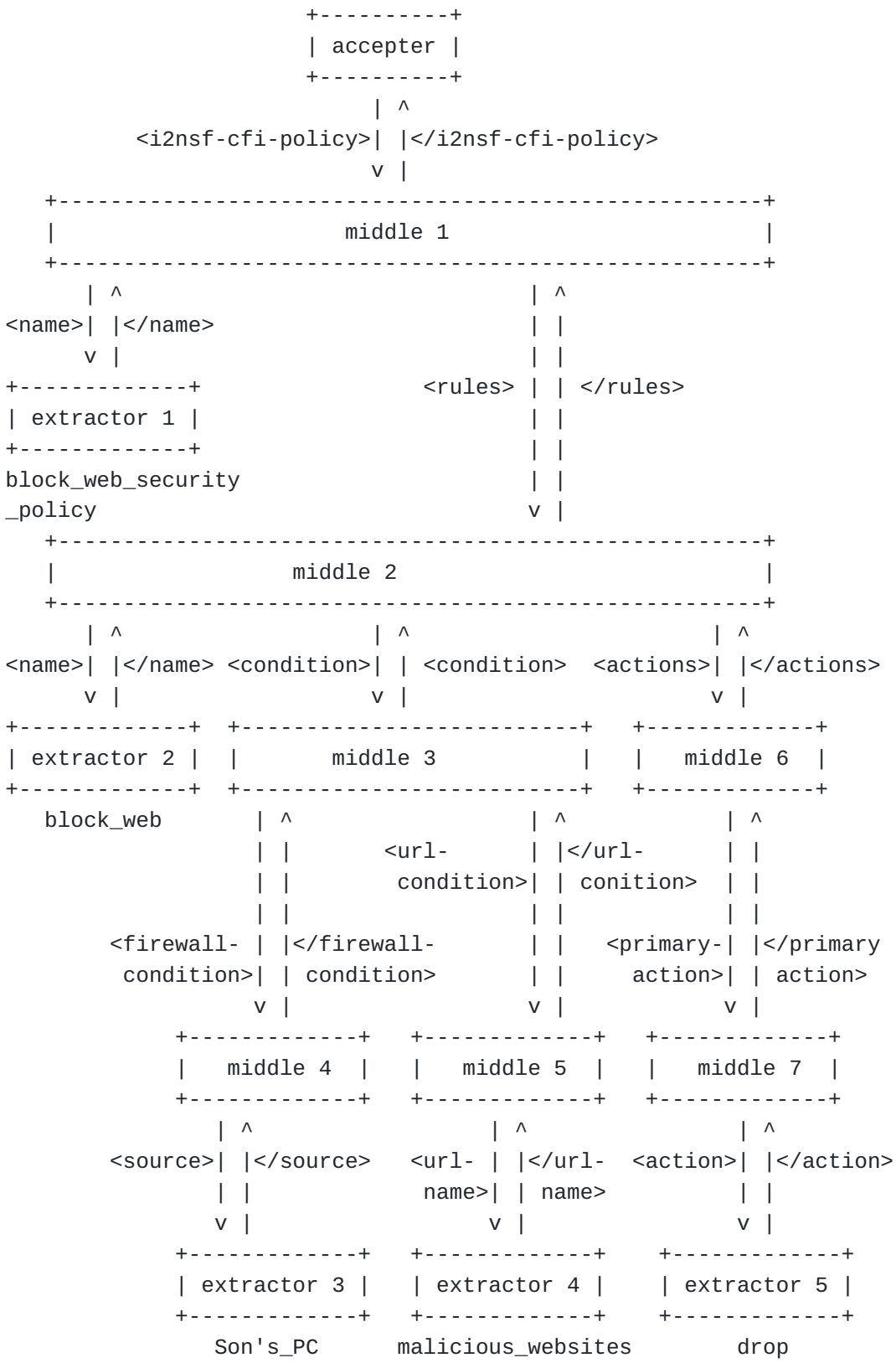


Figure 9: The Example of Data Extractor

To explain the Data Extractor process by referring to an example scenario, assume that Security Controller received a high-level policy for a web-filtering as shown in Figure 8. Then we can construct DFA-based Data Extractor by using the design as shown in Figure 7. Figure 9 shows the architecture of Data Extractor that is based on the architecture in Figure 7 along with the input high-level policy in Figure 8. Data Extractor can automatically extract all of data in the high-level policy according to the following process:

1. Start from the 'accepter' state.

2. Read the first opening tag called '<i2nsf-cfi-policy>', and transit to the 'middle 1' state.

3. Read the second opening tag called '<name>', and transit to the 'extractor 1' state.

4. The current state is an 'extractor' state. Extract the data of 'name' field called 'block_web_security_policy'.

5. Read the second closing tag called '</name>', and go back to the 'middle 1' state.

6. Read the third opening tag called '<rules>', and transit to the 'middle 2' state.

7. Read the fourth opening tag called '<name>', and transit to the 'extractor 2' state.

8. The current state is an 'extractor' state. Extract the data of 'name' field called 'block_web'.

9. Read the fourth closing tag called '</name>', and go back to the 'middle 2' state.

10. Read the fifth opening tag called '<condition>', and transit to the 'middle 3' state.

11. Read the sixth opening tag called '<firewall-condition>', and transit to the 'middle 4' state.

12. Read the seventh opening tag called '<source>', and transit to the 'extractor 3' state.

13. The current state is an 'extractor' state. Extract the data of 'source' field called 'Son's_PC'.

14. Read the seventh closing tag called '</source>', and go back to the 'middle 4' state.

15. Read the sixth closing tag called '</firewall-condition>', and go back to the 'middle 3' state.

16. Read the eight opening tag called '<url-condition>', and transit to the 'middle 5' state.

17. Read the ninth opening tag called '<url-name>', and transit to the 'extractor 4' state.

18. The current state is an 'extractor' state. Extract the data of 'url-name' field called 'malicious_websites'.

19. Read the ninth closing tag called '</url-name>', and go back to the 'middle 5' state.

20. Read the eight closing tag called '</url-condition>', and go back to the 'middle 3' state.

21. Read the fifth closing tag called '</condition>', and go back to the 'middle 2' state.

22. Read the tenth opening tag called '<actions>', and transit to the 'middle 6' state.

23. Read the eleventh opening tag called '<primary-action>', and transit to the 'middle 7' state.

24. Read the twelfth opening tag called '<action>', and transit to the 'extractor 5' state.

25. The current state is an 'extractor' state. Extract the data of 'action' field called 'drop'.

26. Read the twelfth closing tag called '</action>', and go back to the 'middle 7' state.

27. Read the eleventh closing tag called '</primary-action>', and go back to the 'middle 6' state.

28. Read the tenth closing tag called '</actions>', and go back to the 'middle 2' state.

29. Read the third closing tag called '</rules>', and go back to the 'middle 2' state.

30. Read the first closing tag called '</i2nsf-cfi-policy>', and go back to the 'accepter' state.

31. There is no further possible transition, and the state is finally on 'accepter' state. There is no grammar error in Figure 8 so the scanning for data extraction is finished.

The above process is constructed by an extracting algorithm. After finishing all the steps of the above process, Data Extractor can extract all of data in Figure 8, 'block_web_security_policy', 'block_malicious', 'Son's_PC', 'malicious_websites', and 'drop'.

Since the translator is modularized into a DFA structure, a visual understanding is feasible. Also, the performance of Data Extractor is excellent compared to one-to-one searching of data for a particular field. In addition, the management is efficient because the DFA completely follows the hierarchy of Consumer-Facing Interface. If I2NSF User wants to modify the data model of a high-level policy, it only needs to change the connection of the relevant DFA node.

5.3. Data Converter

5.3.1. Role of Data Converter

Every NSF has its own unique capabilities. The capabilities of an NSF are registered into Security Controller by a Developer's Management System, which manages the NSF, via Registration Interface. Therefore, Security Controller already has all information about the capabilities of NSFs. This means that Security Controller can find target NSFs with only the data (e.g., subject and object for a security policy) of the high-level policy by comparing the extracted data with all capabilities of each NSF. This search process for appropriate NSFs is called by policy provisioning, and it eliminates the need for I2NSF User to specify the target NSFs explicitly in a high-level security policy.

Data Converter selects target NSFs and converts the extracted data into the capabilities of selected NSFs. If Security Controller uses this data convertor, it can provide the policy provisioning function to I2NSF User automatically. Thus, the translator design provides big benefits to the I2NSF Framework.

5.3.2. NSF Database

The NSF Database contains all the information needed to convert high-level policy data to low-level policy data. The contents of NSF Database are classified as the following two: "endpoint information" and "NSF capability information".

The first is "endpoint information". Endpoint information is necessary to convert an abstract high-level policy data such as Son's_PC and malicious_websites to a specific low-level policy data

such as 192.0.2.0/24 and malicious1, respectively. In the high-level policy, the range of endpoints for applying security policy MUST be provided abstractly. Thus, endpoint information is needed to specify the abstracted high-level policy data. Endpoint information is provided by I2NSF User as the high-level policy through Consumer-Facing Interface, and Security Controller builds NSF Database based on received information.

The second is "NSF capability information". Since capability is information that allows NSF to know what features it can support, NSF capability information is used in policy provisioning process to search the appropriate NSFs through the security policy. NSF capability information is provided by Developer's Management System (DMS) through Registration Interface, and Security Controller builds NSF Database based on received information. In addition, if the NSF sends monitoring information such as initiating information to Security Controller through NSF-Facing Interface, Security Controller can modify NSF Database accordingly.

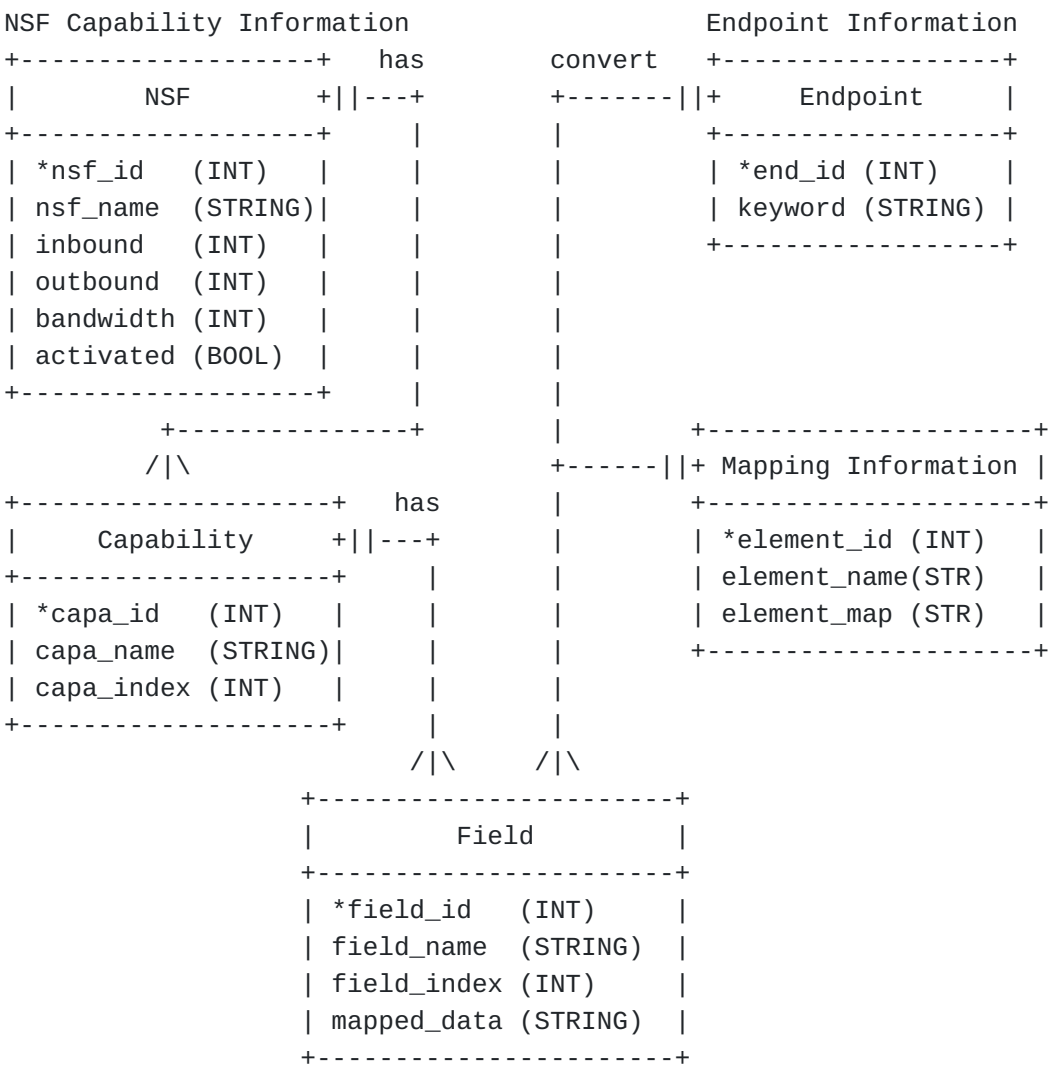


Figure 10: Entity-Relationship Diagram of NSF Database

Figure 10 shows an Entity-Relationship Diagram (ERD) of NSF Database designed to include both endpoint information received from I2NSF User and NSF capability information received from DMS. By designing the NSF database based on the ERD, all the information necessary for security policy translation can be stored, and the network system administrator can manage the NSF database efficiently.

ERD was expressed by using Crow's Foot notation. Crow's Foot notation represents a relationship between entities as a line and represents the cardinality of the relationship as a symbol at both ends of the line. Attributes prefixed with * are key values of each entity. A link with two vertical lines represents one-to-one mapping, and a bird-shaped link represents one-to-many mapping. An NSF entity stores the NSF name (nsf_name), NSF specification (inbound, outbound, bandwidth), and NSF activation (activated). A Capability entity stores the capability name (capa_name) and the index of the capability field in a Registration Interface YANG data model (capa_index). An Endpoint entity stores the keyword of abstract data conversion from I2NSF User (keyword). A Field entity stores the field name (field_name), the index of the field index in an NSF-Facing Interface YANG data model, and converted data by referring to the Endpoint entity and a 'convert' relationship.

5.3.3. Data Conversion in Data Converter

Figure 11: Example of Data Conversion

Figure 11 shows an example for describing a data conversion in Data Converter. High-level policy data MUST be converted into low-level policy data which are compatible with NSFs. If a system administrator attaches a database to Data Converter, it can convert contents by referring to the database with SQL queries. Data conversion in Figure 11 is based on the following list:

*'Policy Name' and 'Rule Name' fields do NOT need the conversion.

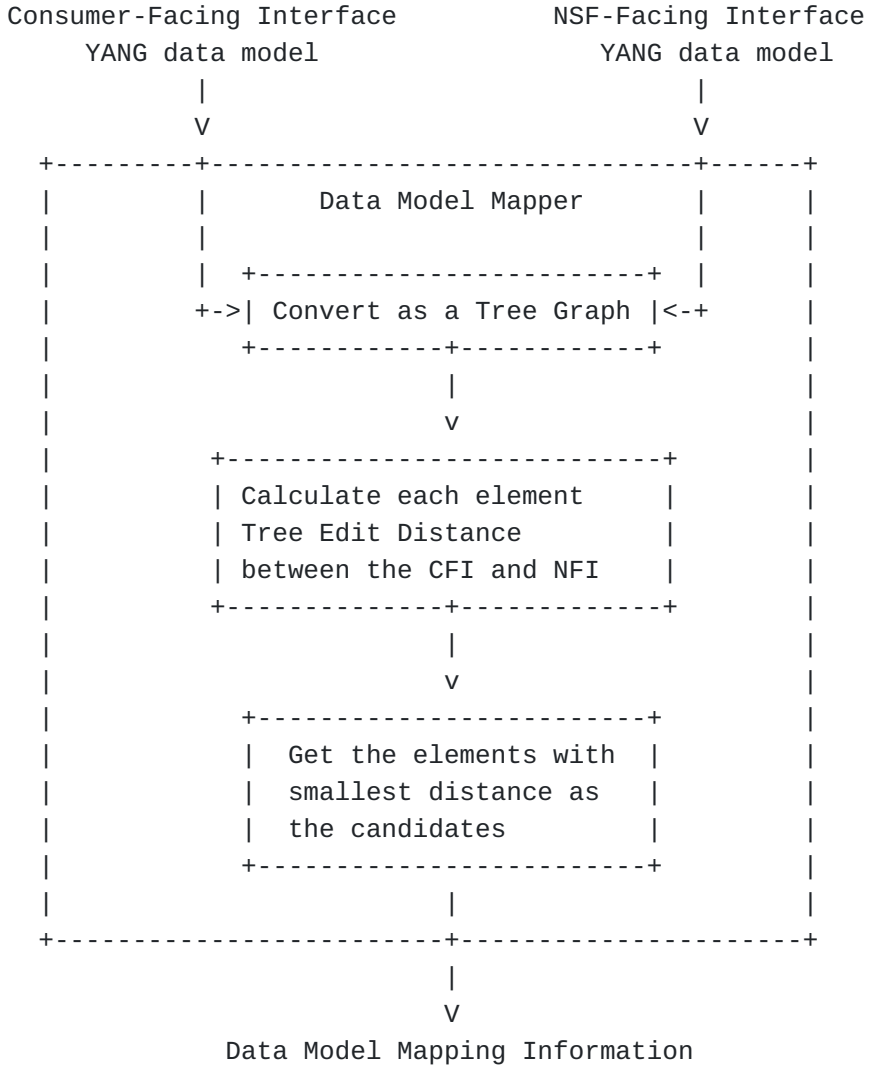
*'Source' field SHOULD be converted into an IPv4 addresses.

*'URL Name' field SHOULD be converted into a URL list of malicious websites.

*'Action' field SHOULD be converted into the corresponding action(s) in NSF capabilities.

5.3.4. Data Model Mapper

When translating a policy, the mapping between each element of the data models are necessary to properly convert the data. The Data Model Mapper create a mapping model between the elements in Consumer-Facing Interface YANG data model and NSF-Facing Interface YANG data model. Each element in the Consumer-Facing Interface Policy Data Model has at least one or more corresponding element in NSF-Facing Interface Data Model.



Note
CFI: Consumer-Facing Interface
NFI: NSF-Facing Interface

Figure 12: Data Model Mapping

Figure 12 shows the automatic mapping method for I2NSF Security Policy Translator. The automatic mapping is helpful as the CFI and NFI YANG data models can be extended. The automatic mapper uses the CFI and NFI YANG data models as inputs. The process the Data Model and converts it into a Tree Graph. Tree Graph is used to proces the Data Model as a Tree instead of individual elements. Then the Data Model Mapper calculates the Tree Edit Distance between each element in Consumer-Facing Interface and each element in NSF-Facing Interface. The Tree Edit Distance can be calculated with an algorithm, e.g., Zhang-Shasha algorithm [Zhang-Shasha], with the calculation should start from the root of the tree.

The Zhang-Shasha calculates the distance by three operations:

*Insert: Inserting a node or element

*Delete: Deleting a node or element

*Change: Change the label of a node or element to another

The insert and delete operations are a simple of adding/deleting a node or element with the length of the label of the node. The change operation must be calculated between the label of the element to produce the distance. There are methods to calculate this, such as Levenshtein Distance, Cosine Similarity, or Sequence Matching. For this data model mapper, cosine similarity should be the best choice as it measures the similarity between words. The data models have similarity between words and it can helps in calculating as minimum distance as possible.

When the minimum distance is obtained, the NSF-Facing Interface element is saved as the candidates for mapping the Consumer-Facing Interface element. This information should be saved to the NSF Database for the Data Converter.

Do note that the proper mapping can be achieved because the similarity between the Consumer-Facing Interface and NSF-Facing Interface. An extension created for the Consumer-Facing Interface and NSF-Facing Interface should keep the close similarity relationship between the data models to be able to produce the mapping model information automatically.

The proper mapping between CFI YANG data model and NFI YANG data model provided in [I-D.ietf-i2nsf-consumer-facing-interface-dm] and [I-D.ietf-i2nsf-nsf-facing-interface-dm], respectively, can be seen in Appendix A.

5.3.4.1. Handling of Default Values for a Low-level Security Policy

Attributes in NFI YANG data model provide detailed configuration for NSFs to handle thorough examination for security services in a network. Some of the attributes in the NFI YANG data model can be given directly after the mapping translation from a high-level policy. But the CFI YANG data model is designed to be used easily by an I2NSF User, hence some attributes cannot be mapped directly to the attributes in the NFI YANG data model.

To accommodate such attributes that cannot be given by the direct translation from the CFI YANG data model, default values can be used. For example, the attribute "default-action" in the NFI YANG data model cannot be configured by the CFI YANG data model. A firewall usually drops packets by default to make sure that only permitted packets are allowed to pass through to the network. So, the default value for the attribute "default-action" will be a "drop" action. This can be done in the implementation of the translation so that the attribute can be given a default value.

The default value for different NSFs can be different depending on the type of service it offers. A typical firewall may use the default-value "drop" for the "default-action" attribute, but an antivirus may use a default-value "pass" for "default-action" to make sure that only the detected viruses are blocked. Other types of firewalls may also use different default values for the "default-action". Thus, the actual default values that are given to the NSFs are out of the scope of this document.

5.3.5. Policy Provisioning

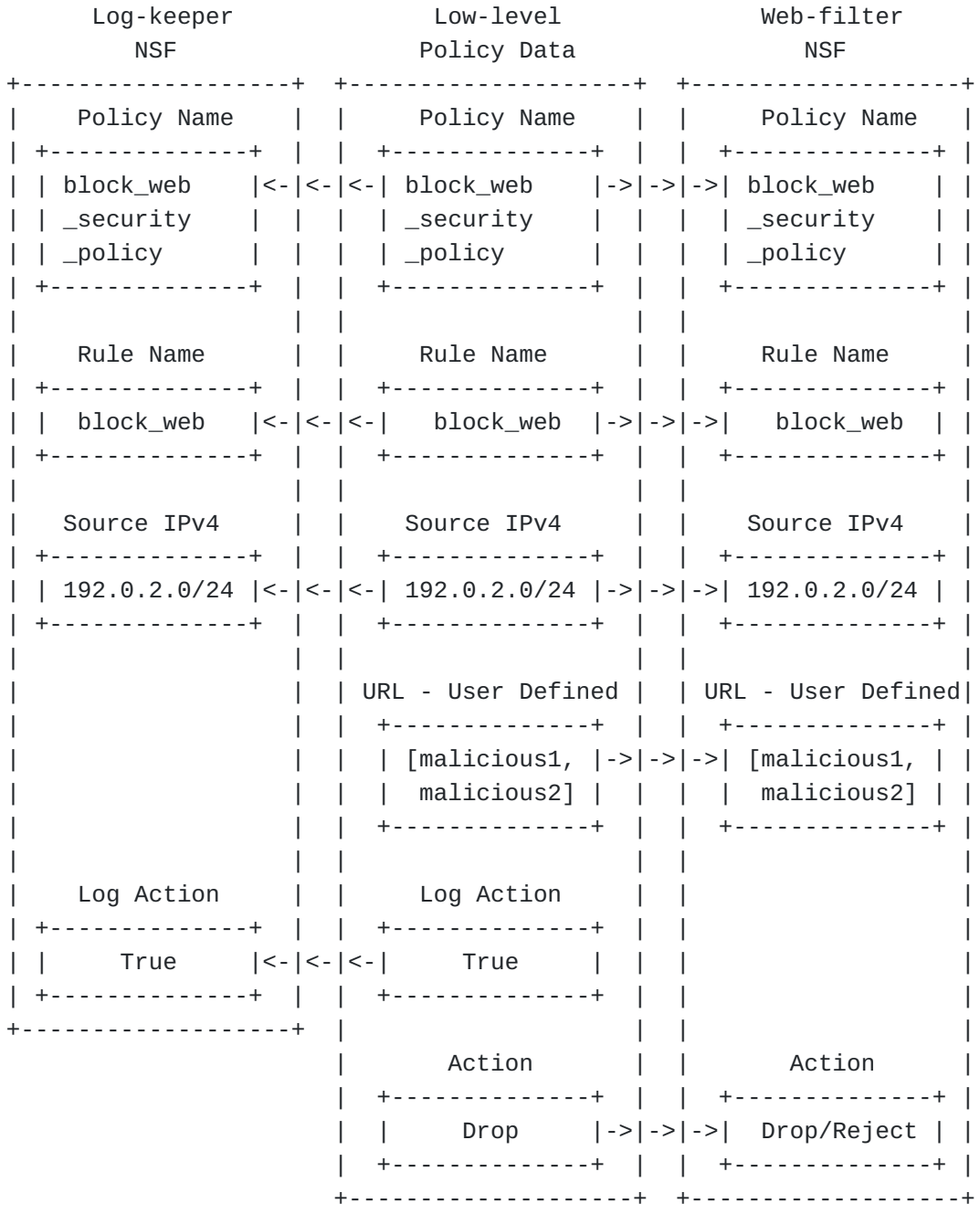


Figure 13: Example of Policy Provisioning

Generator searches for proper NSFs which can cover all of capabilities in the high-level policy. Generator searches for target NSFs by comparing only NSF capabilities which is registered by Vendor Management System. This process is called by "policy provisioning" because Generator finds proper NSFs by using only the policy. If target NSFs are found by using other data which is not included in a user's policy, it means that the user already knows the specific knowledge of an NSF in the I2NSF Framework. Figure 13 shows an example of policy provisioning. In this example, log-keeper NSF and web-filter NSF are selected for covering capabilities in the

security policy. All of capabilities can be covered by two selected NSFs.

5.4. Policy Generator

Generator makes low-level security policies for each target NSF with the extracted data. The low-level security policy can be produced in the form of XML or JSON. Libray such as PyangBind [PyangBind] for Python can be used to parse the NFI YANG data model to produce an XML or JSON form automatically.

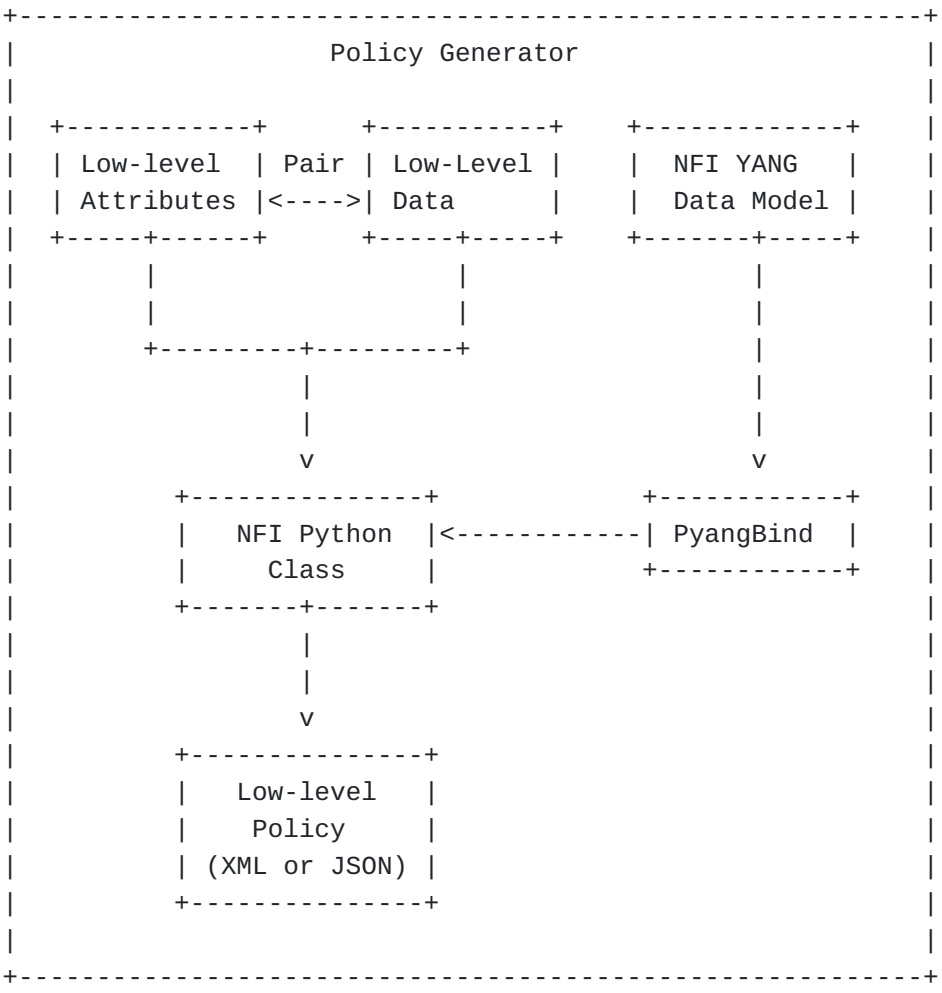


Figure 14: Policy Generator Architecture

Figure 14 shows the architecture of the Policy Generator. First, PyangBind library generates a Python class hierarchy from an input of the NFI YANG data model. This allows low-level data instances from the Data Converter (Section 5.3) to be inserted into the NFI Python Class. To get the appropriate attributes, the low-level data is paired with the attributes received from the Data Model Mapper

(Section 5.3.4). The filled entry can then be encoded into an XML or JSON form automatically by PyangBind.

Figure 15 shows an XML example of a low-level policy generated by the translator.

```
<i2nsf-security-policy>
  <name>block_web_security_policy</name>
  <rules>
    <name>block_web</name>
    <condition>
      <ipv4>
        <ipv4>
          <source-ipv4-network>192.0.2.0/24</source-ipv4-network>
        </ipv4>
      </ipv4>
      <url-category>
        <user-defined>malicious1</user-defined>
        <user-defined>malicious2</user-defined>
      </url-category>
    </condition>
    <action>
      <packet-action>drop</packet-action>
    </action>
  </rules>
</i2nsf-security-policy>
```

Figure 15: Example of Low-Level Policy

6. Implementation Considerations

The implementation considerations in this document include the following three: "data model auto-adaptation", "data conversion", and "policy provisioning".

6.1. Data Model Auto-adaptation

Security Controller which acts as an intermediary entity MUST process the data according to the data model of the connected interfaces. However, the data model can be changed flexibly depending on the situation, and Security Controller may adapt to the change of the data model. Therefore, Security Controller can be implemented for convenience so that the security policy translator can easily adapt to the change of the data model.

The translator constructs and uses the DFA to adapt to the Consumer-Facing Interface Data Model. The DFA starts from the root node of the YANG tree and expands operations by changing the state according to the input. Based on the YANG data model, a container node is

defined as a middle state and a leaf node is defined as an extractor node. After that, if the nodes are connected in the same way as the hierarchical structure of the data model, Security Controller can automatically construct the DFA. Therefore, the DFA can be conveniently built by investigating the link structure using the stack through a Depth-First Search, starting from the root node.

The Policy Generator uses PyangBind to construct the hierarchy of the NFI YANG data model into a Python class. This allows an XML or JSON form to be generated automatically even with updates of the NFI YANG data model. Thus, the security policy translator is able to auto-adapt to the NFI YANG data model.

6.2. Data Conversion

Security Controller requires the ability to materialize the abstract data in the high-level security policy and forward it to NSFs. Security Controller can receive endpoint information as keywords through the high-level security policy. At this time, if the endpoint information corresponding to the keyword is mapped and the query is transmitted to the NSF Database, the NSF Database can be conveniently registered with necessary information for data conversion. When a policy tries to establish a policy through the keyword, Security Controller searches for the details corresponding to the keyword registered in the NSF Database and converts the keyword into appropriate and specific data.

6.3. Policy Provisioning

This document states that a policy provisioning function is necessary to enable an I2NSF User without expert security knowledge to create policies. Policy provisioning is determined by the capability of the NSF. If the information about an NSF's capability for a policy is available to Security Controller, the probability of the selection of an appropriate NSF may increase.

Most importantly, selected NSFs may be able to perform all capabilities in the security policy. This document recommends the study of policy provisioning algorithms that are highly efficient and can satisfy all capabilities in the security policy.

7. Features of Security Policy Translator Design

First, by showing a visualized translator structure, the security manager can handle various policy changes. Translator can be shown by visualizing DFA so that the manager can easily understand the structure of Security Policy Translator.

Second, if it only keeps the hierarchy of the data model, an I2NSF User can freely create high-level policies. In the case of DFA, data

extraction can be performed in the same way even if the order of input is changed. The design of the security policy translator is more flexible than the existing method that works by keeping the tag's position and order exactly.

Third, the structure of Security Policy Translator can be updated even while Security Policy Translator is operating. Because Security Policy Translator is modularized, the translator can adapt to changes in an NSF's capabilities while the I2NSF framework is running. The function of changing the translator's structure can be provided through the Registration Interface [I-D.ietf-i2nsf-registration-interface-dm].

8. Security Considerations

The data saved in the database used by the translation process MUST be kept securely. Some of the data used to convert the data from the high-level security policy to the corresponding low-level security policy may be considered private, e.g., IP addresses mapped to users. And also, alterated data in the database can cause a mistranslation and create a vulnerability in the network security that can be utilized by the attacker.

The configuration and its delivery MUST follow the security considerations discussed in [I-D.ietf-i2nsf-consumer-facing-interface-dm] and [I-D.ietf-i2nsf-nsf-facing-interface-dm].

9. IANA Considerations

This document does not require any IANA actions.

10. References

10.1. Normative References

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol

(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[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>.

[I-D.ietf-i2nsf-consumer-facing-interface-dm] Jeong, J. P., Chung, C., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-consumer-facing-interface-dm-23, 8 August 2022, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-consumer-facing-interface-dm-23.txt>.

[I-D.ietf-i2nsf-nsf-facing-interface-dm] Kim, J. T., Jeong, J. P., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-nsf-facing-interface-dm-29, 1 June 2022, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-nsf-facing-interface-dm-29.txt>.

[I-D.ietf-i2nsf-registration-interface-dm] Hyun, S., Jeong, J. P., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model for NSF Capability Registration", Work in Progress, Internet-Draft, draft-ietf-i2nsf-registration-interface-dm-21, 8 September 2022, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-registration-interface-dm-21.txt>.

[I-D.ietf-i2nsf-capability-data-model] Hares, S., Jeong, J. P., Kim, J. T., Moskowitz, R., and Q. Lin, "I2NSF Capability YANG Data Model", Work in Progress, Internet-Draft, draft-ietf-i2nsf-capability-data-model-32, 23 May 2022, <https://www.ietf.org/archive/id/draft-ietf-i2nsf-capability-data-model-32.txt>.

10.2. Informative References

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

[Automata] Peter, L., "Formal Languages and Automata, 6th Edition", January 2016.

[Zhang-Shasha]
Zhang, K. and D. Shasha, "Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems", SIAM J. Comput. https://www.researchgate.net/publication/220618233_Simple_Fast_Algorithms_for_the_Editing_Distance_Between_Trees_and_Related_Problems, 1989.

[PyangBind] Shakir, R., "PyangBind", PyangBind https://github.com/robshakir/pyangbind, 2018.

[XML] W3C, "On Views and XML (Extensible Markup Language)", June 1999.

[XSLT] W3C, "Extensible Stylesheet Language Transformations (XSLT) Version 1.0", November 1999.

Appendix A. Mapping Information for Data Conversion

Figure 16 shows a mapping list of data fields between Consumer-Facing Interface YANG data model and NSF-Facing Interface YANG data model. Figure 16 describes the process of passing the data value to the appropriate data field of the Data Model in detail after the data conversion.

```
#policy name mapping
/consumer-facing/i2nsf-cfi-policy/name
    -> mapping: /nsf-facing/i2nsf-security-policy
                /name

#rule name mapping
/consumer-facing/i2nsf-cfi-policy/rules/name
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/name

#time mapping
/consumer-facing/i2nsf-cfi-policy/
/rules/event/time/start-date-time
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/event/time/start-date-time

/consumer-facing/i2nsf-cfi-policy/
/rules/event/time/end-date-time
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/event/time/end-date-time

/consumer-facing/i2nsf-cfi-policy/
/rules/event/time/period/day
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/event/time/period/day

/consumer-facing/i2nsf-cfi-policy/
/rules/event/time/period/date
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/event/time/period/date

/consumer-facing/i2nsf-cfi-policy/
/rules/event/time/period/month
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/event/time/period/month

/consumer-facing/i2nsf-cfi-policy/
/rules/event/time/frequency
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/event/time/frequency

#firewall-condition source target reference and mapping
/consumer-facing/i2nsf-cfi-policy/rules/condition
/firewall-condition/source
    -> reference: /consumer-facing/policy
                  /endpoint-group/user-group/name
    -> reference: /consumer-facing/policy
                  /endpoint-group/device-group/name
        -> extract: /consumer-facing/policy
                    /endpoint-group/user-group/mac-address
```

```
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ethernet
                        /source-mac-address
        -> extract: /consumer-facing/policy
                    /endpoint-group/user-group/ip-address
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /source-ipv4-network
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /source-ipv4-range
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv6
                        /source-ipv6-network
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv6
                        /source-ipv6-range

#firewall-condition destination target reference and mapping
/consumer-facing/i2nsf-cfi-policy/rule/condition
/firewall-condition/destination
    -> reference: /consumer-facing/policy
                  /endpoint-group/user-group/name
    -> reference: /consumer-facing/policy
                  /endpoint-group/device-group/name
        -> extract: /consumer-facing/policy
                    /endpoint-group/user-group/mac-address
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ethernet
                        /destination-mac-address
        -> extract: /consumer-facing/policy
                    /endpoint-group/user-group/ip-address
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /destination-ipv4-network
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /destination-ipv4-range
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv6
                        /destination-ipv6-network
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv6
                        /destination-ipv6-range

#ddos-condition threshold mapping
/consumer-facing/i2nsf-cfi-policy/rules/condition
/ddos-condition/packet-rate-threshold
    -> mapping: /nsf-facing/i2nsf-security-policy/rules/condition
```

```
                /ddos/alert-packet-rate

/consumer-facing/i2nsf-cfi-policy/rules/condition
/ddos-condition/packet-byte-threshold
    -> mapping: /nsf-facing/i2nsf-security-policy/rules/condition
                /ddos/alert-byte-rate

/consumer-facing/i2nsf-cfi-policy/rules/condition
/ddos-condition/flow-rate-threshold
    -> mapping: /nsf-facing/i2nsf-security-policy/rules/condition
                /ddos/alert-flow-rate

#payload-condition mapping
/consumer-facing/i2nsf-cfi-policy/rules/condition
/payload-condition/content
    -> reference: /consumer-facing/i2nsf-cfi-policy
                  /threat-prevention/payload-content/name
        -> extract: /consumer-facing/i2nsf-cfi-policy
                    /threat-prevention/payload-content/content
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/payload/content

#voice-condition mapping
/consumer-facing/i2nsf-cfi-policy/rules/condition
/voice-condition/source-id
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/condition/voice
                /source-voice-id

/consumer-facing/i2nsf-cfi-policy/rules/condition
/voice-condition/destination-id
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/condition/voice
                /destination-voice-id

/consumer-facing/i2nsf-cfi-policy/rules/condition
/voice-condition/user-agent
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/condition/voice
                /user-agent

#geographic-location mapping
/consumer-facing/i2nsf-cfi-policy/rules/condition/context
/geographic-location/source
    -> reference: /consumer-facing/i2nsf-cfi-policy
                  /endpoint-groups/location-group/name
        -> extract: /consumer-facing/i2nsf-cfi-policy
                    /endpoint-groups/location-group
                    /geo-ip-ipv4/ipv4-address
```

```
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /source-ipv4-network
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /source-ipv4-range
        -> extract: /consumer-facing/i2nsf-cfi-policy
                    /endpoint-groups/location-group
                    /continent
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/context
                        /geographic-location/source

/consumer-facing/i2nsf-cfi-policy/rules/condition/context
/geographic-location/destination
    -> reference: /consumer-facing/i2nsf-cfi-policy
                  /endpoint-groups/location-group/name
        -> extract: /consumer-facing/i2nsf-cfi-policy
                    /endpoint-groups/location-group
                    /geo-ip-ipv4/ipv4-address
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /destination-ipv4-network
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/ipv4
                        /destination-ipv4-range
        -> extract: /consumer-facing/i2nsf-cfi-policy
                    /endpoint-groups/location-group
                    /continent
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/context
                        /geographic-location/destination

#url-condition mapping
/consumer-facing/i2nsf-cfi-policy/rules/condition
/url-condition/url-name
    -> reference: /consumer-facing/i2nsf-cfi-policy
                  /endpoint-groups/url-group/name
        -> extract: /consumer-facing/i2nsf-cfi-policy
                    /endpoint-groups/url-group/url
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/url-category
                        /pre-defined
            -> mapping: /nsf-facing/i2nsf-security-policy
                        /rules/condition/url-category
                        /user-defined

#rule action name mapping
/consumer-facing/i2nsf-cfi-policy/rules/actions/primary-action
```

```
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/action
                /packet-action/ingress-action
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/action
                /packet-action/egress-action
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/action
                /advanced-action/content-security-control
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/action
                /advanced-action/attack-mitigation-control

/consumer-facing/i2nsf-cfi-policy/rules/actions/secondary-action
    -> mapping: /nsf-facing/i2nsf-security-policy
                /rules/action/packet-action/log-action
```

Figure 16: Mapping Information for Data Conversion

The mapping list shown in the Figure 16 shows all mapped components. This data list should be saved into the NSF Database to provide the mapping information for converting the data. It is important to produce the list automatically as the Consumer-Facing Interface and NSF-Facing Interface can be extended anytime by vendors according to the provided NSF. The Data Model Mapper in Security Policy Translator should be used to produce the mapping model information automatically.

Appendix B. Acknowledgments

This document is a product by the I2NSF Working Group (WG) including WG Chairs (i.e., Linda Dunbar and Yoav Nir) and Diego Lopez. This document took advantage of the review and comments from the following experts: Roman Danyliw and Tom Petch. The authors sincerely appreciate their sincere efforts and kind help.

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea MSIT (Ministry of Science and ICT) (2020-0-00395, Standard Development of Blockchain based Network Management Automation Technology). This work was supported in part by the IITP (R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This work was supported in part by the MSIT under the Information Technology Research Center (ITRC) support program (IITP-2022-2017-0-01633) supervised by the IITP.

Appendix C. Contributors

The following are co-authors of this document:

Chaehong Chung - Department of Electrical and Computer Engineering, Sungkyunkwan University, 2066 Seobu-ro Jangan-gu, Suwon, Gyeonggi-do 16419, Republic of Korea, EMail: darkhong@skku.edu

Jung-Soo Park - Electronics and Telecommunications Research Institute, 218 Gajeong-Ro, Yuseong-Gu, Daejeon, 34129, Republic of Korea, EMail: pjs@etri.re.kr

Younghan Kim - School of Electronic Engineering, Soongsil University, 369, Sangdo-ro, Dongjak-gu, Seoul 06978, Republic of Korea, EMail: younghak@ssu.ac.kr

Appendix D. Changes from draft-yang-i2nsf-security-policy-translation-11

The following changes are made from draft-yang-i2nsf-security-policy-translation-11:

* The XML examples are updated to use the IPv4 address blocks reserved for documentantion, i.e., 10.0.0.0/24 to 192.0.2.0/24.

* The high-level YANG tree is updated using the latest version of [I-D.ietf-i2nsf-consumer-facing-interface-dm].

* The reference of RFC6020 is updated to [RFC7950].

* Section 8 is updated with new security consideration.

Authors' Addresses

Jaehoon Paul Jeong (editor)
Department of Computer Science and Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon
Gyeonggi-Do
16419
Republic of Korea

Phone: +82 31 299 4957
Email: pauljeong@skku.edu
URI: http://iotlab.skku.edu/people-jaehoon-jeong.php

Patrick Lingga
Department of Electronic, Electrical and Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon
Gyeonggi-Do
16419
Republic of Korea

Phone: +82 31 299 4957
Email: patricklink@skku.edu

Jinhyuk Yang
Department of Electronic, Electrical and Computer Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon
Gyeonggi-Do
16419

Republic of Korea

Phone: +82 10 8520 8039
Email: jin.hyuk@skku.edu

Jeonghyeon Kim
Department of Computer Science and Engineering
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon
Gyeonggi-Do
16419
Republic of Korea

Phone: +82 31 299 4957
Email: jeonghyeon12@skku.edu