Network Working Group J. Yang
Internet-Draft J. Jeong
Intended status: Informational J. Kim
Expires: January 3, 2019 Sungkyunkwan University
July 2, 2018

Security Policy Translation in Interface to Network Security Functions
draft-yang-i2nsf-security-policy-translation-00

Abstract

This document proposes a scheme of security policy translation (i.e., Security Policy Translator) 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).

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 3, 2019.

Table of Contents

1. Introduction

This document defines a scheme of a 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 [consumer-facing-inf-dm]. A low-level policy is translated 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 [nsf-facing-inf-dm].

2. Terminology

This document uses the terminology specified in [i2nsf-terminology] [RFC8329].

3. Necessity for 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 [registration-inf-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.

A high-level security policy is specified by RESTCONF/YANG [RFC8040][RFC6020], and a low-level security policy is specified by NETCONF/YANG [RFC6241][RFC6020]. 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 are 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 policy translator in a flexible way while understanding the intention of a policy specification.

This document proposes an approach using Automata theory for the policy tranlation, such as deterministic finite automaton (DFA) and context-free grammar. Note that Automata theory is the foundation of programming languates and compilers. Thus, with this approach, I2NSF User can easily specify a high-level security policy that will be enforced into the corresponding NSFs with a compatible low-level security policy with the help of Policy Translator. Also, for easy managment of Policy Translator, a modularized translator structure is proposed.

4. Design of Policy Translator

Commonly used security policies are created as xml files. A popular way to change the format of an xml file is to use an xslt document. However, the use of xslt makes it difficult to manage the policy translator and to handle the registration of new capabilities of NSFs. With the nessity for policy translator, this document a policy translator based on Automata.

4.1. Overall Structure of Policy Translator

Figure 1 shows the overall design for Policy Translator in Security Controller. There are three main parts for Policy Translator: Extractor, Capability Converter, and Policy Generator.

Extractor is a DFA-based tool for extracting data from a high-level policy which I2NSF User delivered via Consumer-Facing Interface. Data Converter converts the extracted data to the capabilities of target NSFs for a low-level policy. Policy Generator generates a

low-level policy which will execute the NSF capabilities from Converter.

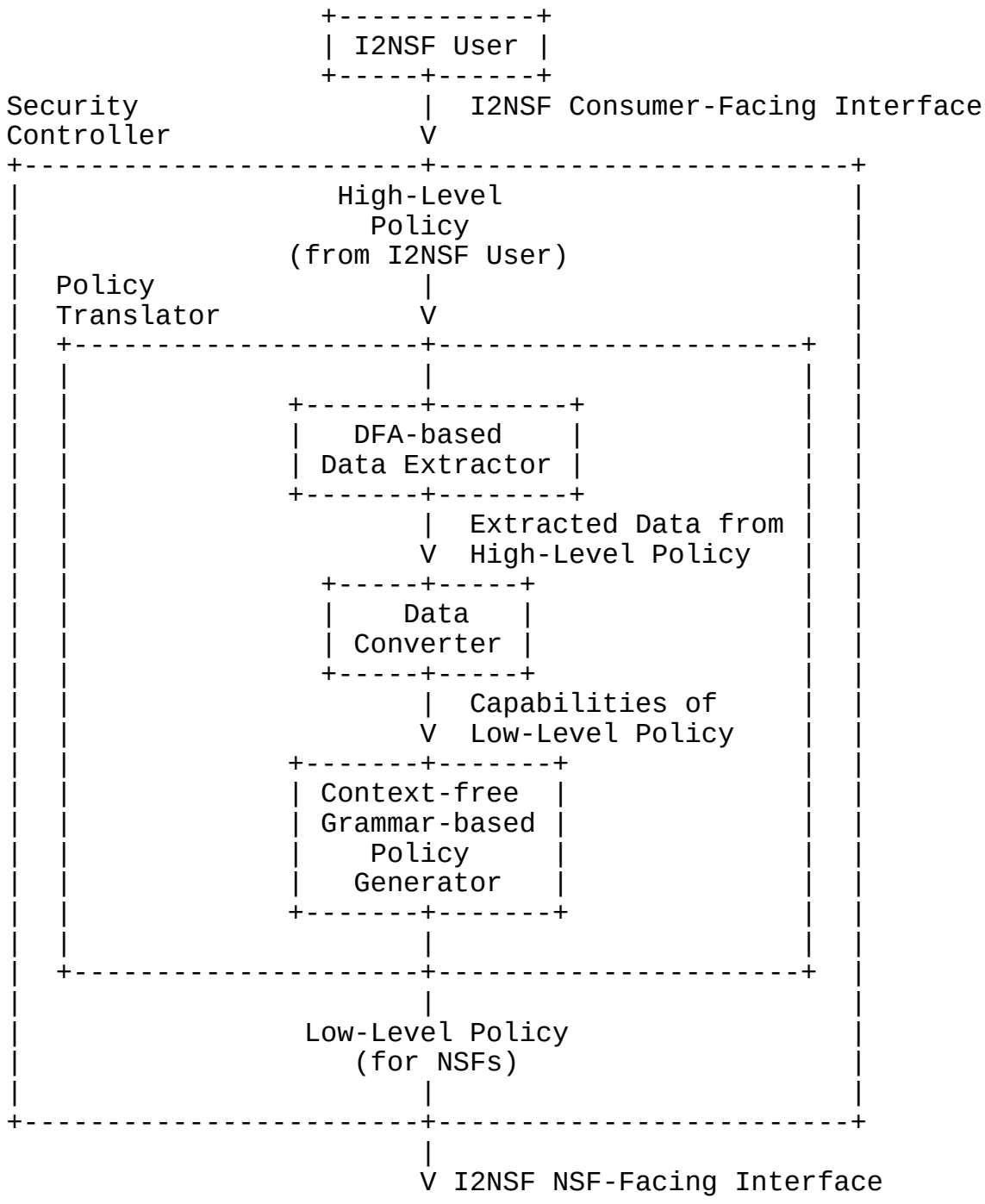

Figure 1: The Overall Design of Policy Translator

4.2. DFA-based Data Extractor

Figure 2 shows a design for Data Extractor in the policy translator. If the high-level policy contains data along the hierarchical structure of the standard YANG data model, data can be easily extracted using the state transition machine, DFA.

The extracted data is 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.

Since the translator is modularized into a DFA structure, a visual understanding is feasible. Also, the following structure of Extractor is easy to manage. 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.

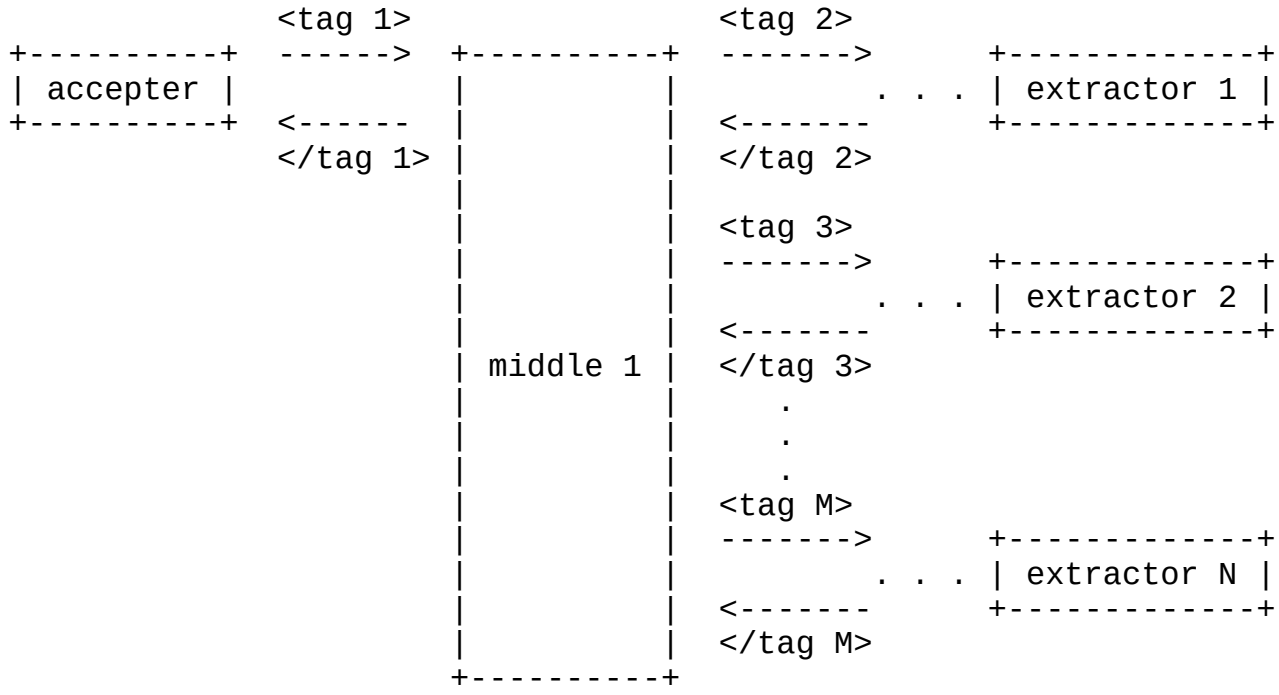

Figure 2: The Design of Data Extractor

4.3. Data Converter

Every NSF has its own unique capability. NSF registration is performed through Registration Interface, and the registration process is dedicated to Security Controller. Therefore, Security Controller already has all information about the capabilities of the NSFs. This means that Security Controller can find target NSFs with only the data of the high-level policy. Data Converter selects

target NSFs and converts the extracted data into the capabilities of selected NSFs. This eliminates the need for I2NSF User to set the target NSFs one by one.

4.4. Context-free Grammar-based Policy Generator

The grammar that makes up the low-level security policy is categorized into two types, Structure Grammar and Content Grammar. Structure Grammar is for grouping other tags into a hierarchy. A security administrator called manager constructs Structure Grammar in the form of an expression as the following equation:

- [structure_grammar] -> <structure_tag>[policy:1][policy:2]...[policy:n]</structure_tag>

Content Grammar is for injecting data into low-level policies to be generated. The manager can construct Content Grammar in the form of an expression as following equation:

- [content_grammar] -> <content_tag>[content_data]</content_tag>

- [content_data] -> data:1 | data:2 | ... | data:n

- [content_grammar] -> [content_grammar][content_grammar] (When duplication is allowed)

5. Features of Policy Translator Design

First, by showing the visualized translator structure, the manager can handle various policy changes. Translator can be shown by visualizing DFA and Context-free Grammar so that the manager can easily understand the structure of Policy Translator.

Second, if I2NSF User only keeps the hierarchy of the data model, 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 policy translator is more flexible than the existing method that works by keeping the tag ’s position and order exactly.

Third, the structure of Policy Translator can be updated even while Policy Translator is operating. Because Policy Translator is modularized, the translator can adapt to changes in the NSF capability while the I2NSF framework is running. The function of changing the translator’s structure can be provided through Registration Interface.

6. Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea MSIT (Ministry of Science and ICT) (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 ITRC (Information Technology Research Center) support program (IITP-2018-2017-0-01633) supervised by the IITP.

7. Informative References

[consumer-facing-inf-dm]
Jeong, J., Kim, E., Ahn, T., Kumar, R., and S. Hares, "I2NSF Consumer-Facing Interface YANG Data Model", draft-ietf-i2nsf-consumer-facing-interface-dm-01 (work in progress), July 2018.

[i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-05 (work in progress), January 2018.

[nsf-facing-inf-dm]
Kim, J., Jeong, J., Park, J., Hares, S., and Q. Lin, "I2NSF Network Security Function-Facing Interface YANG Data Model", draft-ietf-i2nsf-nsf-facing-interface-data-model-01 (work in progress), July 2018.

[registration-inf-dm]
Hyun, S., Jeong, J., Roh, T., Wi, S., and J. Park, "I2NSF Registration Interface YANG Data Model", draft-hyun-i2nsf-registration-dm-04 (work in progress), July 2018.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, February 2018.

Authors' Addresses

Jinhyuk Yang
Department of 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

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

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

Phone: +82 10 8273 0930
EMail: timkim@skku.edu