Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2017

H. Kim
H. Ko
S. Oh
J. Jeong
Sungkyunkwan University
S. Lee
Korea Telecom

July 4, 2016

An Architecture for Security Management in I2NSF Framework
draft-kim-i2nsf-security-management-architecture-01

Abstract

This document describes an architecture for security management in the Interface to Network Security Functions (I2NSF) framework. This security management architecture consists of I2NSF Client, Security Management System (i.e., Security Controller and Developer's Management System), and Network Security Functions (NSFs) in the I2NSF framework. I2NSF Client consists of Application Logic, Policy Updater, and Policy Collector. Security Controller consists of Security Policy Manager and NSF Capability Manager. This document explains their missions and the processing of security management in a high level. It also describes representative use cases, such as security management for the list of malware domains and security management for VoIP-VoLTE.

This Internet-Draft will expire on January 5, 2017.

Copyright Notice

Table of Contents

1. Introduction 3
2. Objectives 3
3. Requirements Language 4
4. Terminology 4
5. Architecture of Security Management 4
5.1. Security Policy Manager 5
5.2. NSF Capability Manager 6
5.3. Developer's Management System 6
5.4. Application Logic 6
5.5. Policy Updater 6
5.6. Policy Collector 6
6. Use Cases 7
6.1. Security Management for the List of Malware Domains 7
6.2. Security Management for VoIP-VoLTE 8
7. Security Considerations 9
8. Acknowledgements 9
9. References 9
9.1. Normative References 9
9.2. Informative References 9

1. Introduction

To enforce a user's high-level security policy into the I2NSF framework [i2nsf-framework], I2NSF Client delivers such a policy to Security Controller via Client Facing Interface. In this document, an architecture for security management is proposed for a given high-level policy in the I2NSF framework. This architecture contains I2NSF Client, Security Management System (i.e., Security Controller and Developer's Management System), and NSFs in the I2NSF framework. I2NSF Client includes Application Logic, Policy Updater, and Policy Collector. Security Controller contains Security Policy Manager and NSF Capability Manager.

Security Policy Manager and NSF Capability Manager in Security Controller are responsible for controlling the updated security policy which will be given by Policy Updater in I2NSF Client via Client Facing Interface. Policy Updater delivers new or updated policies to Security Controller. On the other hand, when an event occurs for NSF to change a low-level policy, Policy Collector receives the correspondingly updated high-level policy via Security Controller. Next, it also updates accordingly the current policies in Application Logic.

In this document, we propose a security management architecture that integrates additional components for security management into the I2NSF framework. Our architecture is designed to support flexible and effective security policies. Application Logic generates the high-level policy and Policy Updater sends it to Security Policy Manager via Client Facing Interface. Security Policy Manager maps the high-level policy into several low-level policies in Security Controller. After mapping into those policies, Security Policy Manager sends them to NSF(s) so that they can be enforced into the NSF(s).

2. Objectives

This document has two main objectives for security management architecture as follows.

- High-level security management: To propose the design of a generic security management architecture to support the enforcement of flexible and effective security policies in NSFs.

- Automatic update of security policies: To provide the reflection of the updated low-level security policies for new security attacks on the corresponding high-level security policies.

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

4. Terminology

This document uses the terminology described in [i2nsf-framework]. In addition, the following terms are defined below:

- Application Logic: It is a component in the security management architecture which generates high-level security policies to block or mitigate security attacks.

- Policy Updater: It is a component which forwards a high-level security policy, which is received from Application Logic, to Security Controller.

- Security Policy Manager: It maps a high-level security policy received from Policy Updater into low-level security policies, and vice verse.

- NSF Capability Manager: It is a component which stores the NSF capability registered by Developer's Management System via Registration Interface and shares it to Security Policy Manger to generate the corresponding low-level security policies.

- Policy Collector: It is a component that forwards the updated high-level security policy to Application Logic.

5. Architecture of Security Management

This section describes a security management architecture in I2NSF and focuses on Security Management System having Security Controller and Developer's Management System. It also explains basic operations of Security Controller. In addition, it describes the details about each component of the architecture.

Figure 1 shows a security management architecture in the I2NSF framework. The architecture is designed to support the enforcement of flexible and effective security policies. Application Logic in I2NSF Client generates a high-level policy in accordance with new security attacks and then Policy Updater in I2NSF Client sends such a policy to Security Policy Manager in Security Controller. Security Policy Manager maps the high-level policy into several low-level policies relevant to NSF capability registered into NSF Capability Manager. After such a mapping into low-level policies, Security

Policy Manager delivers those policies to NSF through NSF Facing Interface. In following sections, we explain the details of each component.

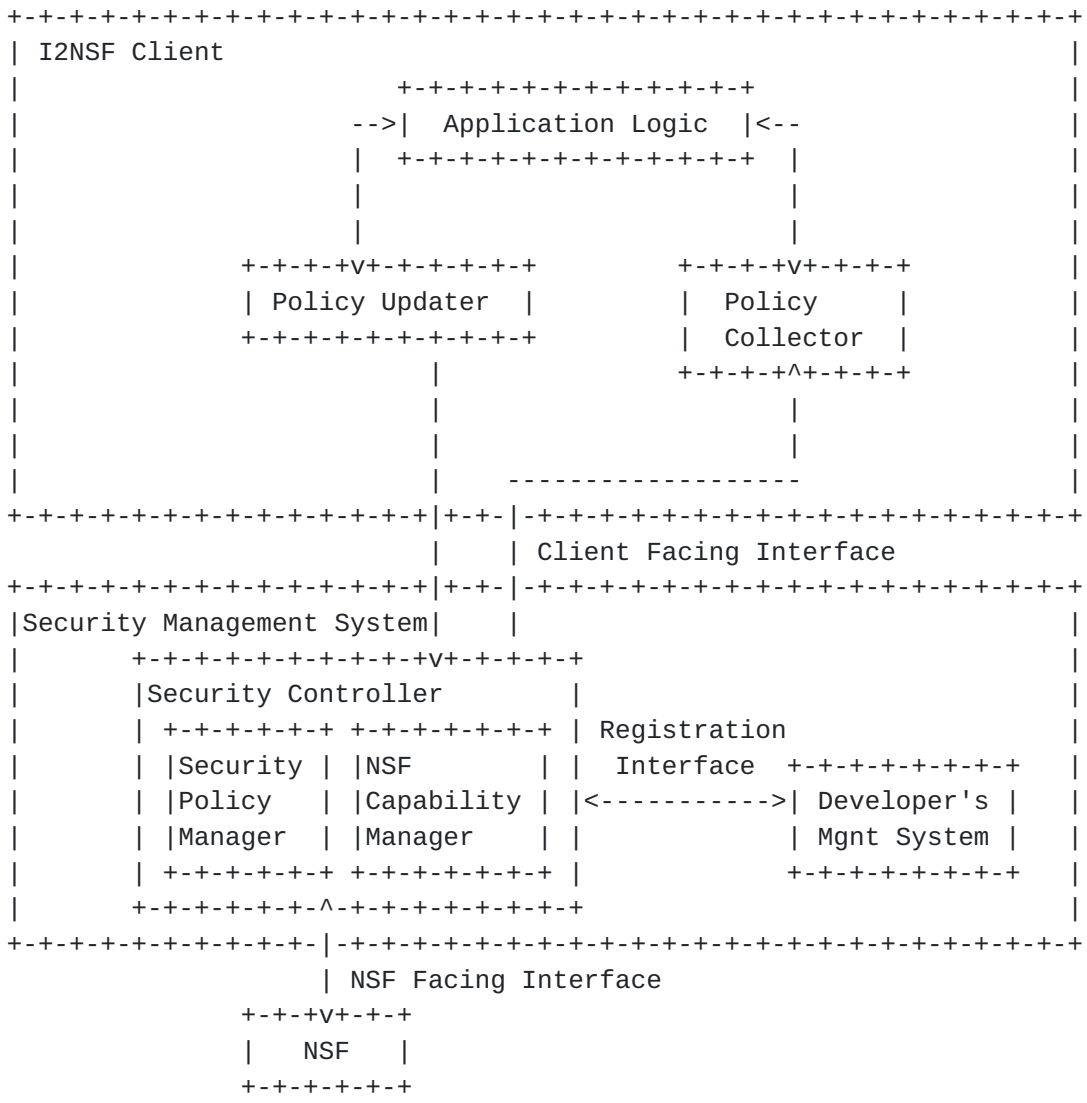

Figure 1: Security Management Architecture in I2NSF

5.1. Security Policy Manager

Security Policy Manager is a component which receives a high-level policy from Policy Updater via Client Facing Interface, and maps the high-level policy into several low-level policies relevant to a given NSF capability from NSF Capability Manager. Moreover, Security Policy Manager delivers those policies to NSF(s) via NSF Facing Interface.

On the other hand, when an event that needs to change the low-level

policy happens in NSF, NSF sends the changed low-level policy to Security Policy Manager via NSF Facing Interface. Security Policy Manager maps such changed low-level policy into the high-level policy and sends it to Policy Collector via Client Facing Interface.

5.2. NSF Capability Manager

NSF Capability Manager is a component integrated into Security Controller. It stores the NSF capability registered by Developer's Management System via Registration Interface and shares it to Security Policy Manager so that Security Policy Manager can generate low-level policies relevant to a given NSF capability. Moreover, whenever a new NSF is registered, NSF Capability Manager requests Developer's Management System to register the NSF capability into the management table of NSF Capability Manager via Registration Interface. On the other hand, when the existing NSF is deleted, NSF Capability Manager eliminates the NSF capability from its management table.

5.3. Developer's Management System

Developer's Management System is a component which registers a new NSF's capability to NSF Capability Manager via Registration Interface. Moreover, in the case where there is some update in the registered NSF, such an update will be delivered from Developer's Management System to NSF Capability Manager.

5.4. Application Logic

Application Logic is a component which generates a high-level security policy to block or mitigate security attacks. It sends the generated policies to Policy Updater. However, this component is out of our standardization scope. We explain its detailed operations in two use cases in Section 6.

5.5. Policy Updater

Policy Updater is a component which receives a high-level security policy generated by Application Logic and delivers it to Security Policy Manager via Client Facing Interface.

5.6. Policy Collector

Policy Collector is a component which receives the updated high level security policy from Security Controller via Client Facing Interface. Such an update is required because the corresponding low-level security policy is updated by some event that occurred in an NSF. After receiving it, Policy Collector forwards it to Application Logic

so that Application Logic can update the corresponding high-level security policy received from Security Controller.

6. Use Cases

A generic architecture is designed to react to possible security attacks. This section shows the procedure of the defense for security attacks in the I2NSF framework [i2nsf-framework] for a given list of security attacks in malware domains and VoIP/VoLTE security attacks.

6.1. Security Management for the List of Malware Domains

Malware domain blacklisting maintains and publishes the blacklists of IP addresses of possible attacking hosts, servers, and networks that are suspicious of malicious activities. Figure 2 shows a security management architecture for Malware Domain Blacklisting.

Based on the malware domain blacklisting, the list of malware domains can be updated either manually or automatically by Malware Domain Manager in I2NSF Client. Also, Malware Domain Manager periodically generates a new high-level security policy to prevent the delivery of packets from/to those newly added malware domains and enforce the low-level security policies in NSF. It sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

An updated low-level policy is sent by an NSF to Security Controller via NSF Facing Interface so that Security Controller can generate the corresponding high-level security policy. Security Controller delivers the high-level security policy to Policy Collector. Policy Collector forwards the policy to Malware Domain Manager as an Application Logic.

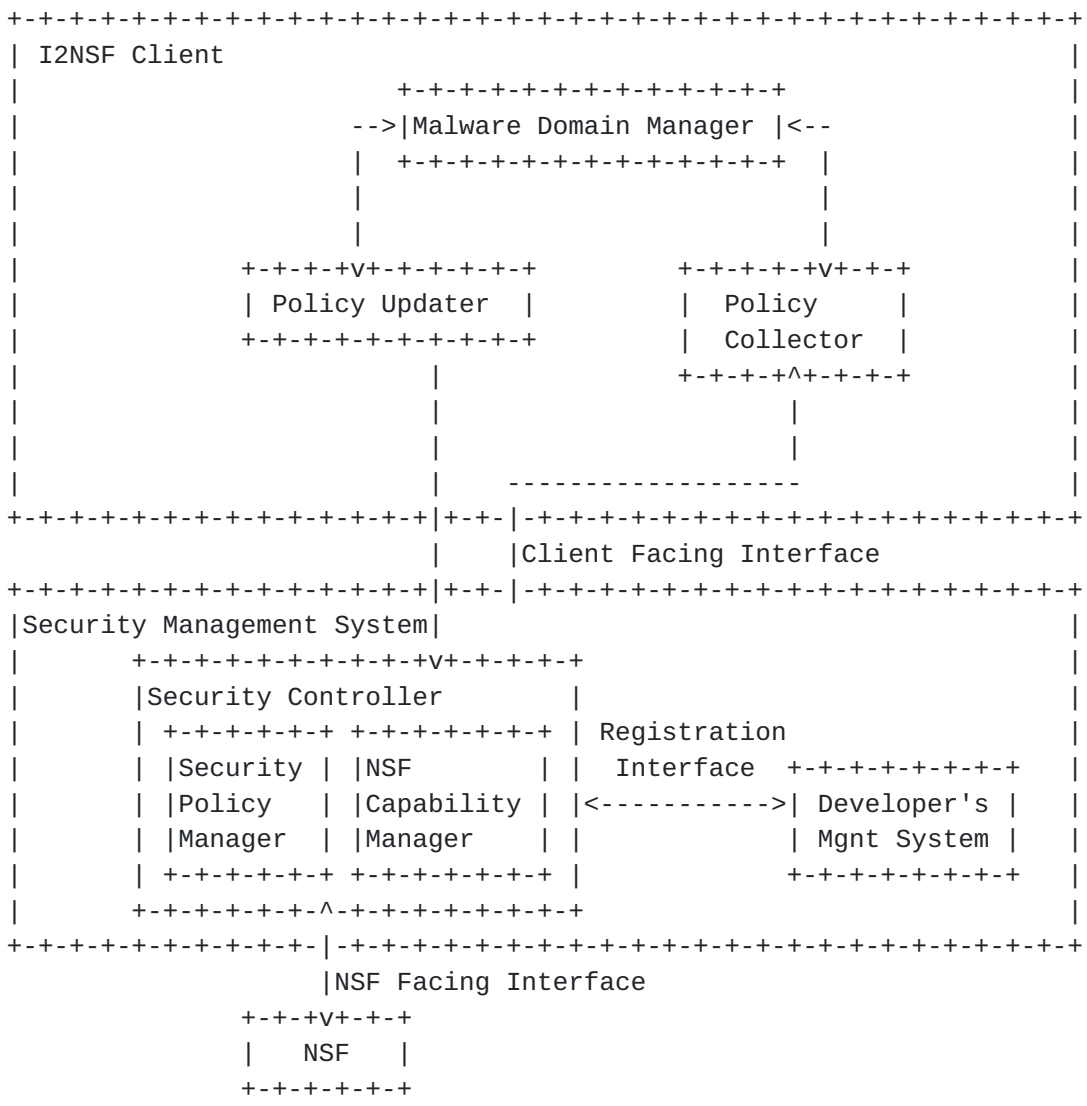

Figure 2: Malware Domain Blacklisting

6.2. Security Management for VoIP-VoLTE

VoIP-VoLTE security management maintains and publishes the blacklists of IP addresses, source ports, expire time, user-agent, and Session Initiation Protocol (SIP) URIs of SIP device that are suspicious of illegal call and authentication. In our generic security management architecture, VoIP-VoLTE Security Manager is Application Logic for VoIP-VoLTE security services in Figure 1.

Based on VoIP-VoLTE security management, the list of illegal devices information can be updated either manually or automatically by VoIP-VoLTE Security Manager as Application Logic. Also, VoIP-VoLTE Security Manager periodically generates a new high-level security policy to prevent the delivery of packets from/to those newly added

VoIP-VoLTE attackers and enforce the low-level security policies in NSF. It sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

An updated low-level policy for VoIP-VoLTE attacks is sent by an NSF to Security Controller via NSF Facing Interface so that Security Controller can generate the corresponding high-level security policy, such as IP addresses, user-agents, and expire time values that need to be added by Security Controller. Security Controller delivers the high-level security policy to Policy Collector. Policy Collector forwards the policy to VoIP-VoLTE Security Manager as an Application Logic.

7. Security Considerations

The security management architecture is derived from the I2NSF framework [i2nsf-framework], so the security considerations of the I2NSF framework should be included in this document. Especially, proper secure communication channels should be used the delivery of control or management messages among the components in the proposed architecture.

8. Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This document has greatly benefited from inputs by Mahdi Daghmehchi-Firoozjaei, Eunsoo Kim, Soyoung Kim, and Tae-Jin Ahn.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[i2nsf-framework] Lopez, E., Lopez, D., Dunbar, L., Strassner, J., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-00 , May 2016.

Authors' Addresses

Hyoungshick Kim
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82 31 299 4324
Fax: +82 31 290 7996
EMail: hyoung@skku.edu
URI: http://seclab.skku.edu/people/hyoungshick-kim/

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

Phone: +82-31-299-4104
EMail: skoh21@skku.edu

Sanghak Oh
Department of Software
Sungkyunkwan University
2066 Seobu-Ro, Jangan-Gu
Suwon, Gyeonggi-Do 16419
Republic of Korea

Phone: +82-31-299-4104
EMail: osh09@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

Se-Hui Lee
Korea Telecom
70 Yuseong-Ro, Yuseong-Gu
Daejeon 305-811
Republic of Korea

Phone: +82 42 870 8162
EMail: sehuilee@kt.com