Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

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

October 31, 2016

An Architecture for Security Management in I2NSF Framework
draft-kim-i2nsf-security-management-architecture-03

Abstract

This document describes an architecture for security management in Interface to Network Security Functions (I2NSF) framework. This security management architecture consists of I2NSF User, Security Management System (i.e., Security Controller and Developer's Management System), and Network Security Functions (NSFs) in the I2NSF framework. I2NSF User consists of Application Logic, Policy Updater, and Event 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, security management for VoIP-VoLTE and time-dependent access control.

Status of This Memo

Table of Contents

1. Introduction

To enforce a user's high-level security policy into the I2NSF framework [i2nsf-framework], I2NSF User delivers such a policy to Security Controller via Consumer-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 User, Security Management System (i.e., Security Controller and Developer's Management System), and NSFs in the I2NSF framework. I2NSF User includes Application Logic, Policy Updater, and Event 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 User via Consumer-Facing Interface. If a new policy were created or existing policies needed to be updated, Policy Updater delivers them to Security Controller. On the other hand, when an event occurs for NSF to change a low-level policy, NSF sends the event to Security Controller. Security Controller then forwards it to Event Collector. Next, Event Collector sends it to Application Logic. Application Logic then updates the current policies in accordance with the event.

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 a high-level policy and Policy Updater sends it to Security Policy Manager via Consumer-Facing Interface. Security Policy Manager maps the high-level policy into several low-level policies in Security Controller. After mapping, the low-level policies are distributed to NSF(s) so that they can be enforced in them.

2. 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].

3. Terminology

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

o 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 to Security Controller. The high-level policy is received from Application Logic.

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

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

- Event Collector: It is a component which receives an event which should be reflected on updating (or generating) a high-level policy in Application Logic, from Security Controller.

4. Objectives

The two main objectives for security management architecture in this document are 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 an event which occurs for NSFs to change a low-level policy for new security attacks on the corresponding high-level security policies.

5. Architecture of Security Management

This section describes a security management architecture in I2NSF and focuses on Security Management System containing Security Controller and Developer's Management System. It also explains some basic operations of Security Controller and describes the details of each component consisting 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 User generates a high-level policy in accordance with new security attacks, and Policy Updater in I2NSF User then sends the policy to Security Policy Manager in Security Controller. Security Policy Manager maps the high-level policy into several low-level policies which are relevant to NSF capability registered into NSF Capability Manager. After mapping, the low-level policies are

distributed to NSF(s) by Security Policy Manager through NSF-Facing Interface. In the following sections, we explain the details of each component.

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | I2NSF User                                                        |
   |                        +-+-+-+-+-+-+-+-+-+-+-+                    |
   |                     ---|  Application Logic  |<--                 |
   |                     |  +-+-+-+-+-+-+-+-+-+-+-+  |                 |
   |                     |                           |                 |
   |                     |                           |                 |
   |              +-+-+-+v+-+-+-+-+-+         +-+-+-+v+-+-+-+          |
   |              | Policy Updater  |         |  Event      |          |
   |              +-+-+-+-+-+-+-+-+-+         |  Collector  |          |
   |                          |               +-+-+-+^+-+-+-+          |
   |                          |                      |                 |
   |                          |                      |                 |
   |                          |    -------------------                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+|+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                              |    | Consumer-Facing Interface
   +-+-+-+-+-+-+-+-+-+-+-+-+-+|+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Security Management System|    |                                   |
   |       +-+-+-+-+-+-+-+-+-+-+v+-+-+-+-+                               |
   |       |Security Controller        |                               |
   |       | +-+-+-+-+-+ +-+-+-+-+-+-+ | Registration                  |
   |       | |Security | |NSF        | |  Interface  +-+-+-+-+-+-+-+   |
   |       | |Policy   | |Capability | |<----------->| Developer's |   |
   |       | |Manager  | |Manager    | |             | Mgnt System |   |
   |       | +-+-+-+-+-+ +-+-+-+-+-+-+ |             +-+-+-+-+-+-+-+   |
   |       +-+-+-+-+-+-^-+-+-+-+-+-+-+-+                               |
   +-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                       | NSF-Facing Interface
                  +-+-+v+-+-+
                  |   NSF   |
                  +-+-+-+-+-+
```

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 Consumer-Facing Interface, and maps the high-level policy into several low-level policies which are relevant to a given NSF capability from NSF Capability Manager. Additionally, Security Policy Manager delivers those policies to NSF(s) via NSF-Facing Interface.

On the other hand, when an event that requires the low-level policies

to be changed happens in NSF, NSF sends the event to Security Policy Manager via NSF-Facing Interface. Security Policy Manager then sends it to Event collector via Consumer-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 with 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 case there are some updates in the registered NSF, such updates 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, and 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 Consumer-Facing Interface.

5.6. Event Collector

Event Collector receives an event, which should be reflected on updating (or generating) a high-level policy in Application Logic, from Security Controller. The procedure of receiving an event in NSF is necessary because a low-level security policy can be updated according to a specific event that occurred in an NSF. After receiving the event, Event Collector forwards it to Application Logic

so that Application Logic can update (or generate) a high-level security policy based on the event received from Security Controller.

6. Use Cases

A generic architecture is designed to react to security attacks that can occur in a real world environment. 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, VoIP/VoLTE security attacks, and time-dependent access control.

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 User. Malware Domain Manager also 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. Additionally, Malware Domain Manager sends the new high-level security policy to Policy Updater, which forwards it to Security Controller.

When NSF detects a new dangerous domain, the corresponding IP addresses are sent by an NSF to Security controller via NSF-Facing Interface. Security Controller delivers the IP addresses to Event Collector, which in turn forwards the IP addresses to Dangerous Domain Manager. Finally, Dangerous Domain Manager updates the Dangerous domain database.

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | I2NSF User                                                        |
   |                        +-+-+-+-+-+-+-+-+-+-+-+-+                  |
   |                     ---|Malware Domain Manager |<--               |
   |                     |  +-+-+-+-+-+-+-+-+-+-+-+-+  |               |
   |                     |                             |               |
   |                     |                             |               |
   |              +-+-+-+v+-+-+-+-+-+         +-+-+-+-+v+-+-+          |
   |              | Policy Updater  |         |  Event      |          |
   |              +-+-+-+-+-+-+-+-+-+         |  Collector  |          |
   |                          |               +-+-+-+^+-+-+-+          |
   |                          |                      |                 |
   |                          |                      |                 |
   |                          |    -------------------                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+|+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                              |    |Consumer-Facing Interface
   +-+-+-+-+-+-+-+-+-+-+-+-+-+|+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Security Management System|    |                                   |
   |       +-+-+-+-+-+-+-+-+-+v+-+-+-+-+                               |
   |       |Security Controller        |                               |
   |       | +-+-+-+-+-+ +-+-+-+-+-+-+ | Registration                  |
   |       | |Security | |NSF        | |  Interface  +-+-+-+-+-+-+-+   |
   |       | |Policy   | |Capability | |<----------->| Developer's |   |
   |       | |Manager  | |Manager    | |             | Mgnt System |   |
   |       | +-+-+-+-+-+ +-+-+-+-+-+-+ |             +-+-+-+-+-+-+-+   |
   |       +-+-+-+-+-+-^-+-+-+-+-+-+-+-+                               |
   +-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                       |NSF-Facing Interface
                  +-+-+v+-+-+
                  |   NSF   |
                  +-+-+-+-+-+
```

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-agents, 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 plays the role of 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.

When the NSF detects an anomalous message or call delivered from a domain, the domain information such as an IP address, user-agents and expire time values is sent by an NSF to Security Controller via NSF-Facing Interface. Security Controller delivers it to Event Collector. Event Collector forwards the detected domain information to VoIP-VoLTE Security Manager, and then VoIP-VoLTE Security Manager updates the VoIP-VoLTE database.

6.3. Security Management for Time-Dependent Access Control

Time-dependent access control policies manage a user's access to particular websites during a certain period of time. For example, in a company, a manager blocks employees' access to Youtube, which is a big distraction during working hours.

Based on time-dependent access control, I2NSF User registers the list of blocked websites and blocking time at Application logic. Application logic stores the list into database and generates a high-level security policy (e.g., blocking the access to websites by checking the blocked websites and blocking time in the list). Application logic delivers it to Policy updater, and then Policy updater forwards it to Security controller. In Security controller, Security policy manager maps the high-level policy to low-level policies, and then it sends the corresponding NSFs to enforce the low-level policies.

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 for the delivery of control or management messages amongst 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, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-04 (work in progress), October 2016.

Appendix A. Changes from draft-kim-i2nsf-security-management-architecture-02

The following changes were made from draft-kim-i2nsf-security-management-architecture-02:

o This version reflects the framework for I2NSF in draft-ietf-i2nsf-framework-04 with the updated terminology.

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