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

S. Hyun
S. Woo
Y. Yeo
J. Jeong
Sungkyunkwan University
J. Park
ETRI

July 4, 2016

Service Function Chaining-Enabled I2NSF Architecture
draft-hyun-i2nsf-sfc-enabled-i2nsf-00

Abstract

This document describes an architecture of the I2NSF framework using security function chaining for securiy policy enforcement. Security function chaining enables composite inspection of network traffic by steering the traffic through multiple types of security functions according to the information model for the capability layer interface in the I2NSF framework. This document explains the additional components integrated into the I2NSF framework and their functionalities to achieve security function chaining. It also describes representative use cases to address major benefits from the proposed architecture.

Copyright Notice

Table of Contents

1. Introduction   3
2. Objective   3
3. Terminology   4
4. Architecture   6
  4.1. SFC Policy Manager   8
  4.2. SFC Catalog Manager   8
  4.3. Developer's Management System   9
  4.4. Classifier   9
  4.5. Service Function Forwarder (SFF)   9
5. Use Cases 10
  5.1. Dynamic Path Alternation 10
  5.2. Enforcing Different SFPs Depending on Trust Levels . . . 11
  5.3. Effective Load Balancing with Dynamic SF Instantiation . 12
6. Security Considerations 13
7. Acknowledgements 13
8. References 13
  8.1. Normative References 13
  8.2. Informative References 14

1. Introduction

To effectively cope with emerging sophisticated network attacks, it is necessary that various security functions cooperatively analyze network traffic [sfc-ns-use-cases] [RFC7498] [i2nsf-problem-and-use-cases] [i2nsf-capability-interface-im]. In addition, depending on the characteristics of network traffic and their suspiciousness level, the different types of network traffic need to be analyzed through the different sets of security functions. In order to meet such requirements, besides security policy rules for individual security functions, we need an additional policy about service function chaining (SFC) for network security [sfc-ns-use-cases] which determines a set of security functions through which network traffic packets should pass for inspection. In addition, [i2nsf-capability-interface-im] proposes an information model for capability layer interface of the I2NSF framework that enables a security function to trigger further inspection by executing additional security functions based on its own analysis results [i2nsf-framework]. However, the current design of the I2NSF framework does not consider network traffic steering fully in order to enable such chaining between security functions.

In this document, we propose an architecture that integrates additional components for security function chaining into the I2NSF framework. We extend the security controller's functionalities such that it can interpret a high-level policy of security function chaining into a low-level policy and manage them. It also keeps the track of the available service function (SF) instances for security functions and their information (e.g., network information and workload), and makes a decision on which SF instances to use for a given security function chain/path. Based on the forwarding information provided by the security controller, the service function forwarder (SFF) performs network traffic steering through various required security functions. A classifier is deployed for the enforcement of SFC policies given by the security controller. It performs traffic classification based on the polices so that the traffic passes through the required security function chain/path by the SFF.

2. Objective

o Policy configuration for security function chaining: SFC-enabled I2NSF architecture allows policy configuration and management of security function chaining. Based on the chaining policy, relevant network traffic can be analyzed through various security functions in a composite, cooperative manner.

- o Network traffic steering for security function chaining: SFC-enabled I2NSF architecture allows network traffic to be steered through multiple required security functions based on the SFC policy. Moreover, the I2NSF information model for capability layer interface [i2nsf-capability-interface-im] requires a security function to call another security function for further inspection based on its own inspection result. To meet this requirement, SFC-enabled I2NSF architecture also enables traffic forwarding from one security function to another security function.

- o Load balancing over security function instances: SFC-enabled I2NSF architecture provides load balancing of incoming traffic over available security function instances by leveraging the flexible traffic steering mechanism. For this objective, it also performs dynamic instantiation of a security function when there are an excessive amount of requests for that security function.

3. Terminology

This document uses the terminology described in [RFC7665], [RFC7665] [sfc-ns-use-cases] [i2nsf-terminology][ONF-SFC-Architecture].

- o Service Function/Security Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers) [RFC7665]. In this document, SF is used to represent both Service Function and Security Function. Sample Security Service Functions are as follows: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.

- o Classifier: An element that performs Classification. It uses a given policy from SFC Policy Manager.

- o Service Function Chain (SFC): A service function chain defines an ordered set of abstract service functions and ordering constraints that must be applied to packets and/or frames and/or flows selected as a result of classification [RFC7665].

- o Service Function Forwarder (SFF): A service function forwarder is responsible for forwarding traffic to one or more connected service functions according to information carried in the SFC encapsulation, as well as handling traffic coming back from the SF. Additionally, an SFF is responsible for delivering traffic to

a classifier when needed and supported, transporting traffic to another SFF (in the same or the different type of overlay), and terminating the Service Function Path (SFP) [RFC7665].

- Service Function Path (SFP): The service function path is a constrained specification of where packets assigned to a certain service function path must be forwarded. While it may be so constrained as to identify the exact locations for packet processing, it can also be less specific for such locations [RFC7665].

- SFC Policy Manager: It is responsible for translating a high-level policy into a low-level policy, and performing the configuration for SFC-aware nodes, passing the translated policy and configuration to SFC-aware nodes, and maintaining a stabilized network.

- SFC Catalog Manager: It is responsible for keeping the track of the information of available SF instances. For example, the information includes the supported transport protocols, IP addresses, and locations for the SF instances.

- Control Nodes: It collectively refer to SFC Policy Manager, SFC Catalog Manager, SFF, and Classifier.

- Service Path Identifier (SPI): It identifies a service path. The classifier MUST use this identifier for path selection and the Control Nodes MUST use this identifier to find the next hop [sfc-nsh].

- Service Index (SI): It provides a location within the service path. SI MUST be decremented by service functions or proxy nodes after performing the required services [sfc-nsh].

- Network Service Header (NSH): The header is used to carry SFC related information. Basically, SPI and SI should be conveyed to the Control Nodes of SFC via this header.

- SF Forwarding Table: SFC Policy Manager maintains this table. It contains all the forwarding information on SFC-enabled I2NSF architecture. Each entry includes SFF identifier, SPI, SI, and next hop information. For example, an entry ("SFF: 1", "SPI: 1", "SI: 1", "IP: 192.168.xx.xx") is interpreted as follows: "SFF 1" should forword the traffic containing "SPI 1" and "SI 1" to "IP=192.168.xx.xx".

4. Architecture

This section describes an SFC-enabled I2NSF architecture and the basic operations of service chaining. It also includes details about each component of the architecture.

Figure 1 describes the components of SFC-enabled I2NSF architecture. Our architecture is designed to support a composite inspection of traffic packets in transit. According to the inspection result of each SF, the traffic packets could be steered to another SF for futher detailed analysis. It is also possible to reflect a high-level SFC-related policy and a configuration from I2NSF Client on the components of the original I2NSF framwork. Moreover, the proposed architecture provides load balancing, auto supplementary SF generation, and the elimination of unused SFs. In order to achieve these design purposes, we integrate several components to the original I2NSF framwork. In the following sections, we explain the details of each component.

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | I2NSF Client                                                      |
   |              +-+-+-+-+-+-+-+-+                                    |
   |              |Client/        |                                    |
   |              |App Controller |                                    |
   |              +-+-+-+^+-+-+-+-+                                    |
   |                     |                                             |
   |                     |                                             |
   +-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                         | Client Facing Interface
                         | (called Service Layer Interface)
   +-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Security Management System                                        |
   |                     |                                             |
   |       +-+-+-+-+-+-+-v-+-+-+-+-+                                   |
   |       |Security Controller    |                                   |
   |       | +-+-+-+-+  +-+-+-+-+  | Registration                      |
   |       | |SFC    |  |SFC    |  |   Interface +-+-+-+-++-+-+-+-+    |
   |       | |Policy |  |Catalog|  |<----------->| Developer's    |    |
   |       | |Manager|  |Manager|  |             | Mgnt System    |<-+ |
   |       | +-+-+-+-+  +-+-+-+-+  |             +-+-+-+-++-+-+-+-+  | |
   |       +-+-+-+-+-+-+-^-+-+-+-+-+-+         Instance Layer Interface| |
   +-+-+-+-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                       | NSF Facing Interface                         |
                       | (called Capability Layer Interface)          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Security Network   |                                             | |
   |       +------------------------------------------------+        | |
   |       |                    |                           |        | |
   |       |              +-+-+-v-+-+-+            +-+-+-+-+v+-+-+   | |
   | +-+-+-v-++-+         |  +-+-+-+  |            |    +-+-+-+  |   | |
   | |          |         |  |SFF1 |<-|---------------->| SF1 |  |   | |
   | |Classifier|<------> |  +-+-+-+< |            |    +-+-+-+  |   | |
   | |          |         |          \|            |    +-+-+-+  |   | |
   | +-+-+-+-++-+         |  +-+-+-+  \---------------->| SF2 |  |<--- |
   |                      |  |SFF2 |<-|---------------/ +-+-+-+  |     |
   |                      |  +-+-+-+<-|-------------\   +-+-+-+  |     |
   |                      +-+-+-+-+-+-+              \->| SF3 |  |     |
   |                                               |    +-+-+-+  |     |
   |                                               +-+-+-+-+-+-+-+     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: SFC-enabled I2NSF

4.1. SFC Policy Manager

SFC Policy Manager is a core component in our system. It is responsible for the following two things: (1) Interpreting a high-level SFC policy (or configuration) into a low-level SFC policy (or configuration), which is given by I2NSF Client, and delivering the interpreted policy to Classifiers for security function chaining. (2) Generating an SF forwarding table and distributing the fowarding information to SFF(s) by consulting with SFC Catalog Manager. As Figure 1 describes, SFC Policy Manager performs these additional functionalities through Service Layer Interface and Capability Layer Interface.

Given a high-level SFC policy/configuration from I2NSF Client via Service Layer Interface, SFC Policy Manager interprets it into a low-level policy/configuration comprehensible to Classifier(s), and then delivers the resulting low-level policy to them. Moreover, SFC Policy Manager possibly generates new policies for the flexible change of traffic steering to rapidly react to the current status of SFs. For instance, it could generate new rules to forward all subsequent packets to "Firewall Instance 2" instead of "Firewall Instance 1" in the case where "Firewall Instance 1" is under congestion.

SFC Policy Manager gets information about SFs from SFC Catalog Manager to generate SF forwarding table. In the table generation process, SFC Policy Manager considers various criteria such as SFC policies, SF load status, SF physical location, and supported transport protocols. An entry of the SF forwarding table consists of SFF Identifier, SFP, SI, and next hop information. The examples of next hop information includes the IP address and supported transport protocols (e.g., VxLAN and GRE). These forwarding table updates are distributed to SFFs with either push or pull methods.

4.2. SFC Catalog Manager

In Figure 1, SFC Catalog Manager is a component integrated into Security Controller. It is responsible for the following three things: (1) Maintaining the information of every available SF instance such as IP address, supported transport protocol, service name, and load status. (2) Responding the queries of available SF instances from SFC Policy Manager so as to help to generate a forwarding table entry relevant to a given SFP. (3) Requesting Developer's Management System for the dynamic instantiation of supplementary SF instances to avoid service congestion or the elimination of an existing SF instance to avoid resource waste.

Whenever a new SF instance is registered, Developer's Management

System passes the information of the registered SF instance to SFC Catalog Manager, so Catalog Manager maintains a list of the information of every available SF instance. Once receiving a query of a certain SFP from SFC Policy Manager, SFC Catalog Manager searches for all the available SF instances applicable for that SFP and then returns the search result to SFC Policy Manager.

In our system, each SF instance periodically reports its load status to SFC Catalog Manager. Based on such reports, SFC Catalog Manager updates the information of the SF instances and manages the pool of SF instances by requesting Developer's Management System for the additional instantiation or elimination of the SF instances. Consequently, SFC Catalog Manager enables efficient resource utilization by avoiding congestion and resource waste.

4.3. Developer's Management System

We extend Developer's Management System for additional functionalities as follows. As mentioned above, SFC Catalog Manager requests Developer's Management System to create additional SF instances when the existing instances of that service function are congested. On the other hand, when there are an excessive number of instances for a certain service function, SFC Policy Manager requests Developer's Management System to eliminate some of the SF instances. As a response to such requests, Developer's Management System creates and/or removes SF instances. Once it creates a new SF instance or removes an existing SF instance, the changes must be notified to SFC Catalog Manager.

4.4. Classifier

Classifier is a logical component that may exist as a standalone component or a submodule of another component. In our system, the initial classifier is typically located at an entry point like a border router of the network domain, and performs the initial classification of all incoming packets according to the SFC policies, which are given by SFC policy manager. The classification means determining the SFP through which a given packet should pass. Once the SFP is decided, the classifier constructs an NSH that specifies the corresponding SPI and SI, and attaches it to the packet. The packet will then be forwarded through the determined SFP on the basis of the NSH information.

4.5. Service Function Forwarder (SFF)

It is responsible for the following two functionalities: (1) Forwarding the packets to the next SFF/SF. (2) Handling re-classification request from SF.

An SFF basically takes forwarding functionality, so it needs to find the next SF/SFF for the incoming traffic. It will search its forwarding table to find the next hop information that corresponds to the given traffic. In the case where the SFF finds a target entry on its forwarding table, it just forwards the traffic to the next SF/SFF specified in the next hop information. If an SFF does not have an entry for a given packet, it will request the next hop information to SFC Policy Manager with SFF identifier, SPI, and SI information. The SFC Policy Manager will respond to the SFF with next hop information, and then the SFF updates its forwarding table with the response, forwarding the traffic to the next hop.

Sometimes an SF may want to forward the traffic, which is highly suspicious, to another SF for futher inspection. The SF then appends the inspection result to the MetaData field of the NSH and delivers it to the source SFF. The attached MetaData includes a re-classification request to change the SFP of the traffic to another SFP for stronger inspection. When the SFF receives the traffic requiring re-classification, it forwards the traffic to the Classifier where re-classification will be eventually performed.

5. Use Cases

This section introduces three use cases for the SFC-enabled I2NSF architecture : (1) Dynamic Path Alternation, (2) Enforcing Different SFPs Depending on Trust Levels, and (3) Effective Load Balancing with Dynamic SF Instantiation.

5.1. Dynamic Path Alternation

In SFC-enabled I2NSF architecture, a Classifier determines the initial SFP of incoming traffic according to the SFC policies. The classifier then attaches an NSH specifying the determined SFP of the packets, and they are analyzed through the SFs of the initial SFP. However, SFP is not a static property, so it could be changed dynamically through re-classification. A typical example is for a certain SF in the initial SFP to detect that the traffic is highly suspicious (likely to be malicious). In this case, the traffic needs to take stronger inspection through a different SFP which consists of more sophisticated SFs.

Figure 2 illustrates an example of such dynamic SFP alternation in a DDoS attack scenario. SFP-1 represents the default Service Function Path that the traffic initially follows, and SFP-1 consists of AVC, Firewall, and IDS/IPS. If the IDS/IPS suspects that the traffic is attempting DDoS attacks, it will change the SFP of the traffic from the default to SFP-2 so that the DDoS attack mitigator can execute a proper countermeasure against the attack.

Such SFP alternation is possible in the proposed architecture with re-classification. In Figure 1, to initiate re-classification, the IDS/IPS appends its own inspection result to the MetaData field of NSH and deliver it to the SFF from which it has originally received the traffic. The SFF then forwards the received traffic including the inspection result from the IDS/IPS to Classifier for re-classification. Classifier checks the inspection result and determines the new SFP (SFP-2) associated with the inspection result in the SFC policy, and updates the NSH with the SPI of SFP-2. The traffic is forwarded to the DDoS attack mitigator.

```
                SFP-1. AVC:Firewall:IDS/IPS
 ------------------------------------------------------------------>
 +-+-+-+-+   +-+-+-+   +-+-+-+-+-+-+     +-+-+-+-+-+   +-+-+-+-+-+-+-+-+
 | Source|---| AVC |---| Firewall|-----| IDS/IPS |---| Destination |
 +-+-+-+-+   +-+-+-+   +-+-+-+-+-+-+     +-+-+-+-+-+   +-+-+-+-+-+-+-+-+
 ----------------------------------------,                 ,------>
                                          \    +-+-+-+-+   /
                                           \   |  DDoS |  /
                                            \  +-+-+-+-+ /
                                             '----------'
                                  SFP-2. AVC:Firewall:DDoS:IDS/IPS
```

Figure 2: Dynamic SFP Alternation Example

5.2. Enforcing Different SFPs Depending on Trust Levels

Because the traffic coming from a trusted source is highly likely to be harmless, it does not need to be inspected excessively. On the other hand, the traffic coming from an untrusted source requires an in-depth inspection. By applying minimum required security functions to the traffic from a trusted source, it is possible to prevent the unnecessary waste of resources. In addition, we can concentrate more resources on potential malicious traffic. In the SFC-enabled I2NSF architecture, by configuring an SFC Policy to take into account the levels of trust of traffic sources, we can apply different SFPs to the traffic coming from different sources.

Figure 3(a) and Figure 3(b) represent SFPs applicable to traffic from trusted and untrusted sources, respectively. In Figure 3(a), we assume a lightweight IDS/IPS which is configured to perform packet header inspection only. In this scenario, when receiving the traffic from a trusted source, the classifier determines the SFP in Figure 3(a) such that the traffic passes through just a simple analysis by the lightweight IDS/IPS. On the other hand, traffic from an untrusted source passes more thorough examination through the SFP

in Figure 3(b) which consists of three different types of SFs.

```
+-+-+-+-+-+            +-+-+-+-+-+            +-+-+-+-+-+-+-+
| Source  |----------->| IDS/IPS |----------->| Destination |
+-+-+-+-+-+            +-+-+-+-+-+            +-+-+-+-+-+-+-+

          (a) Traffic flow of trusted source



+-+-+-+-+-+            +-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+
| Source  |            | Anti-Spoofing |      | Destination |
+-+-+-+-+-+            | function      |      +-+-+-+^+-+-+-+
    |                  +-+^+-+-+-+-+-+-+             |
    |                     |         |                |
    |       +-+-+-+-+-+-+ |         |    +-+-+-+-+-+ |
    ------->| Firewall  |--         ---->|   DPI   |--
            +-+-+-+-+-+-+                +-+-+-+-+-+


          (b) Traffic flow of untrusted source
```

Figure 3: Different path allocation depending on source of traffic

5.3. Effective Load Balancing with Dynamic SF Instantiation

In a large-scale network domain, there typically exist a large number of SF instances that provide various security services. It is possible that a specific SF instance experiences an excessive amount of traffic beyond its capacity. In this case, it is required to allocate some of the traffic to another available instance of the same security function. If there are no additional instances of the same security function available, we need to create a new SF instance and then direct the subsequent traffic to the new instance. In this way, we can avoid service congestion and achieve more efficient resource utilization. This process is commonly called load balancing.

In the SFC-enabled I2NSF architecture, SFC Catalog Manager performs periodic monitoring of the load status of available SF instances. In addition, it is possible to dynamically generate a new SF instance through Developer's Management System. With these functionalities along with the flexible traffic steering mechanism, we can eventually provide load balancing service.

The following describes the detailed process of load balancing when congestion occurs at the firewall instance:

1. SFC Catalog Manager detects that the firewall instance is receiving too much requests. Currently, there are no additional firewall instances available.

2. SFC Catalog Manager requests Developer's Management System to create a new firewall instance.

3. Developer's Management System creates a new firewall instance and then registers the information of the new firewall instance to SFC Catalog Manager.

4. SFC Catalog Manager updates the SFC Information Table to reflect the new firewall instance, and notifies SFC Policy Manager of this update.

5. Based on the received information, SFC Policy Manager updates the forwarding information for traffic steering and sends the new forwarding information to the SFF.

6. According to the new forwarding information, the SFF forwards the subsequent traffic to the new firewall instance. As a result, we can effecively alleviate the burden of the existing firewall instance.

6. Security Considerations

To enable security function chaining in the I2NSF framework, we adopt the additional components in the SFC architecture. Thus, this document shares the security considerations of the SFC architecture that are specified in [RFC7665] for the purpose of achieving secure communication among components in the proposed architecture.

7. 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).

8. References

8.1. Normative References

[RFC7665] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7665, March 2014.

[sfc-nsh] Lopez, E., Lopez, D., Dunbar, L., Zhuang, X., Parrott, J., Krishnan, R., and S. Durbha, "Framework for Interface to Network Security Functions", draft-ietf-sfc-nsh-05 , June 2015.

[sfc-ns-use-cases] Wang, E., Leung, K., Felix, J., and J. Iyer, "Service Function Chaining Use Cases for Network Security", draft-wang-sfc-ns-use-cases-01 , March 2016.

8.2. Informative References

[RFC7498] Quinn, P. and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, April 2015.

[i2nsf-capability-interface-im] Xia, L., Zhang, D., Lopez, E., Bouthors, N., and L. Fang, "Information Model of Interface to Network Security Functions Capability Interface", draft-xia-i2nsf-capability-interface-im-05 , March 2016.

[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.

[i2nsf-problem-and-use-cases] Hares, S., Dunbar, L., Lopez, D., Zarny, M., and C. Jacquenet, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-00 , February 2016.

[i2nsf-terminology] Hares, S., Strassner, J., Lopez, D., and L. Xia, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-00 , April 2016.

[ONF-SFC-Architecture]   ONF, "L4-L7 Service Function Chaining Solution Architecture", June 2015.

Authors' Addresses

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: swhyun77@skku.edu
URI: http://imtl.skku.ac.kr/

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: suwoo@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

Phone: +82 31 290 7222
Fax: +82 31 299 6673
EMail: yunsuk@imtl.skku.ac.kr,
URI: http://imtl.skku.ac.kr/index.php?mid=member_student

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

Jung-Soo Park
Electronics and Telecommunications Research Institute
218 Gajeong-Ro, Yuseong-Gu
Daejeon 305-700
Republic of Korea

Phone: +82 42 860 6514
EMail: pjs@etri.re.kr