An Automata-based Security Policy Translation for Network Security Functions

Jinhyuk Yang*, Jaehoon (Paul) Jeong†
* Department of Electrical and Computer Engineering, Sungkyunkwan University, Republic of Korea
† Department of Interaction Science, Sungkyunkwan University, Republic of Korea
Email: {jin.hyuk, pauljeong}@skku.edu

***Abstract*—This paper proposes the design of a security policy translator in Interface to Network Security Functions (I2NSF) framework. Also, this paper shows the benefits of designing security policy translations. I2NSF is an architecture for providing various Network Security Functions (NSFs) to users.**
I2NSF user should be able to use NSF even if user has no overall knowledge of NSFs. Generally, policies which are generated by I2NSF user contain abstract data because users do not consider the attributes of NSFs when creating policies. Therefore, the I2NSF framework requires a translator that automatically finds the NSFs which is required for policy when Security Controller receives a security policy from the user and translates it for selected NSFs. We satisfied the above requirements by modularizing the translator through Automata theory.

Keywords*—Policy Translation, Automata theory, Interface to Network Security Functions

I. Introduction

Until now, the cases in which an individual or a company's system has been lax, causing a security attack and damaging it, have been steadily occurring. This is a problem that cannot be ignored, and many users want to keep the system secure. However, according to CIO, nearly half of the companies do not have officers who specialize in security and manage it [1]. There are many users who want to keep the security of the system like this, but administrators who know security professionally are pretty rare.

To solve this problem, Interface to Network Security Functions (I2NSF) Framework [2] was proposed. Internet Engineering Task Force (IETF) [3] has formed a working group to study I2NSF and is actively working on standardization. I2NSF [4] is an architecture for providing various Network Security Functions (NSFs) such as firewall and web filter to users. The goal of the I2NSF is to provide a unified interface that allows users who require system security to use appropriate NSFs according to their desired policies. .

Generally, NSFs should be designed so that users can get services without the professional involvement of NSF, because security users do not know the expertise of NSF. To do this, I2NSF need a policy translator that helps users set up NSF policies without expert knowledge. In this paper, we propose the new design of the security policy translator.

We constructed the policy translator by using Automata theory for convenient management. First, we made Extractor for extracting data from the high-level policy by using Deterministic Finite Automaton. (DFA) Second, we attached NSF database to Data Converter for data mapping from abstract data to specified data which is required for NSF. Last, we constructed Generator for generating the low-level policies for each target NSF by using Context-free Grammar.

The rest of this paper is organized as follows. Section II shows the necessity of our proposed translator design. Section III explains the I2NSF framework to help to understand the paper. Section IV describes our proposed design of security policy translator based on Automata theory. In Section V, we describe the implementation of our proposed design. We finally conclude the paper with future work in Section VI.

II. Necessity of Policy Translator

This section shows the necessity of our proposed policy translator with examples. The following two sentences are examples of policies that block certain malicious websites:

- Block my son's computers from malicious websites.
- Drop packets from the IP address 10.0.0.1 and 10.0.0.3 to www.malicious.com and www.illegal.com

The above two examples are sentences for the same operation. However, NSF cannot understand the policy of the first sentence. The Web-filter NSF that supports the ability to block Web sites must receive at least the source IP address and Web site address for operation. In the first sentence, NSF cannot operate normally because the IP address and the address of the malicious website are not specified correctly.

Conversely, when a user decides to make a policy for security, the user never drops packet like the second example. The second sentence can be generated if the user is using web-filter NSF for policy setting and knows that they need to pass the source IP address and website address. It means that the user understands the company's NSF professionally, but there are not many users available.

In conclusion, the I2NSF user will issue the policy in the first sentence, and NSF will require the same policy as the second sentence. The policies that users create contain abstract data and can be generated without any special knowledge of NSF, and it is called by high-level policy. The policy required by the NSF contains detailed data and should be able to set all of the required capabilities for operation and it is called by low-level policy.

I2NSF Framework must translate user created high-level policy into low-level policy in order to achieve goal. Also, I2NSF should be able to find proper NSFs automatically for applying the high-level policy. In this paper, we propose an Automata-based security policy translator to solve two requirements.

III. Background

This section describes background knowledge of I2NSF Framework to help to understand the paper. We explain the entities of I2NSF Framework and process of applying policies.

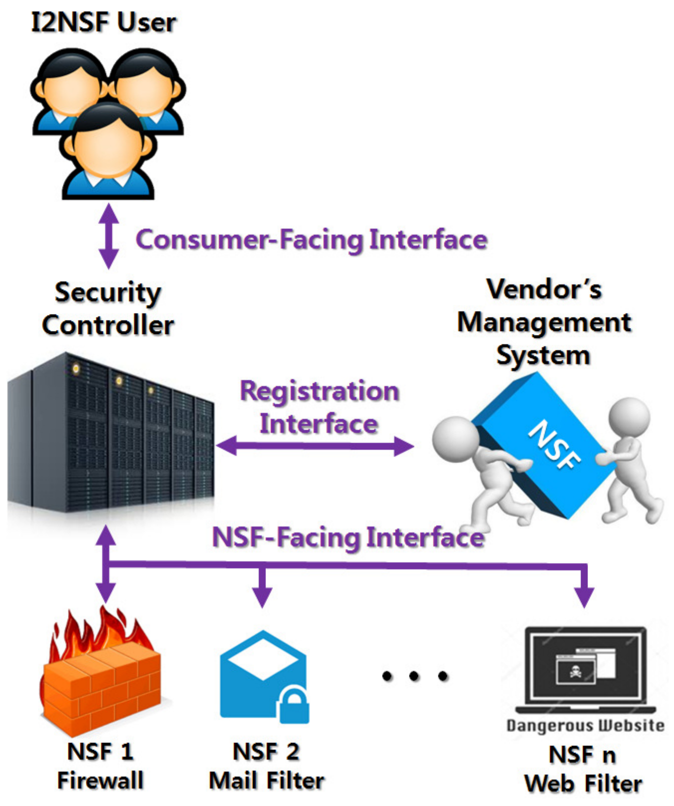
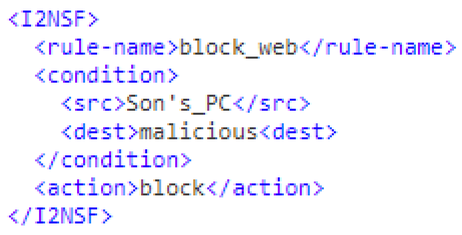


Fig. 1. Architecture of I2NSF framework



```
<I2NSF>
  <rule-name>block_web</rule-name>
  <condition>
    <src>Son's_PC</src>
    <dest>malicious<dest>
  </condition>
  <action>block</action>
</I2NSF>
```

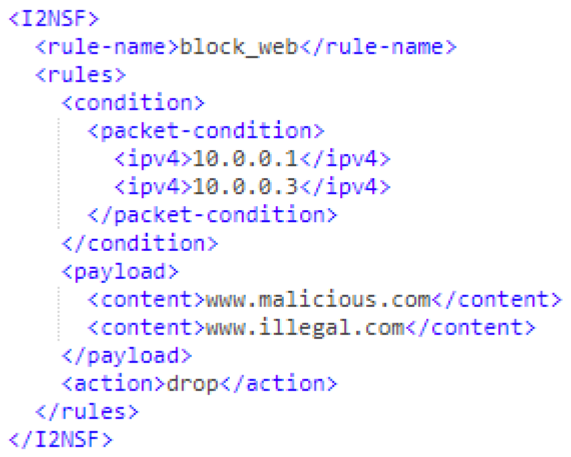
Fig. 2. Example of high-level policy

A. Architecture of I2NSF Framework

This subsection describes the entities of I2NSF architecture. Figure 1 illustrates I2NSF architecture which is including three main interfaces.

Vendor's Management System, which is utilized by a service provider, installs package of NSFs via Registration Interface. Then, registered NSFs can provide security functions to I2NSF User. I2NSF Users can generate the high-level policy (e.g., Figure 2) to keep security for their network system. XML format is generally used for policy expression. Security Controller translates high-level policy to low-level policy (e.g., Figure 3). Also, Security Controller should search proper NSFs for covering the high-level policy requirement.

Consume-Facing Interface which is between I2NSF User and Security Controller allows the client to communicate with Security Controller using the RESTCONF [5] protocol and YANG [7] data model. NSF-Facing Interface which is between Security Controller and NSFs transfers rule for specific NSF and allows Security Controller to communicate with NSFs using the NETCONF [6] protocol and YANG data model.



```
<I2NSF>
  <rule-name>block_web</rule-name>
  <rules>
    <condition>
      <packet-condition>
        <ipv4>10.0.0.1</ipv4>
        <ipv4>10.0.0.3</ipv4>
      </packet-condition>
    </condition>
    <payload>
      <content>www.malicious.com</content>
      <content>www.illegal.com</content>
    </payload>
    <action>drop</action>
  </rules>
</I2NSF>
```

Fig. 3. Example of low-level policy

Registration Interface which is between Security Controller and Vendor's Management System registers NSFs with their capabilities [8] using the NETCONF protocol and YANG data model. Capability is the ability for operation, and Security Controller can search proper NSFs for I2NSF User's policy by comparing capabilities of registered NSFs.

B. Process of Applying Policy

This subsection describes how to create and apply policy in I2NSF Framework (Figure 1) as follows:

1) Vendor's Management System registers all capabilities of provided NSFs to Security Controller.
2) I2NSF User makes high-level policy and sends it to Security Controller using Consumer-Facing Interface.
3) Security Controller searches target NSFs, which is able to cover the high-level policy, by comparing with the capabilities of registered NSFs.
4) Security Controller translates received high-level policy to low-level policy for target NSF. Then, sends it to target NSF via NSF-Facing Interface.
5) The target NSFs are configured by each received low-level policy.

IV. Security Policy Translator

This section explains our proposed design of Automata-based security policy translator in detail. Figure 4 shows the overall architecture of security policy translator. When I2NSF User generate a high-level policy and send it to Security Controller, the policy translator in Security Controller can find the target NSFs and translate the high-level policy to low-level policies for each target NSF as follows:

1) When I2NSF User send the high-level policy, Extractor in policy translator extracts the data of the high-level policy by using Deterministic Finite Automaton (DFA).
2) Extracted data is converted to NSF required data via Data Converter. Data Convert can convert to data, which is proper with NSF, by comparing with NSF database.

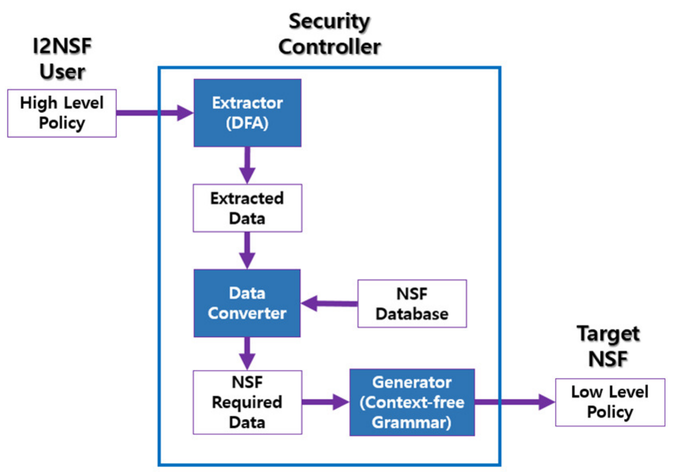


Fig. 4. Overall architecture of policy translator

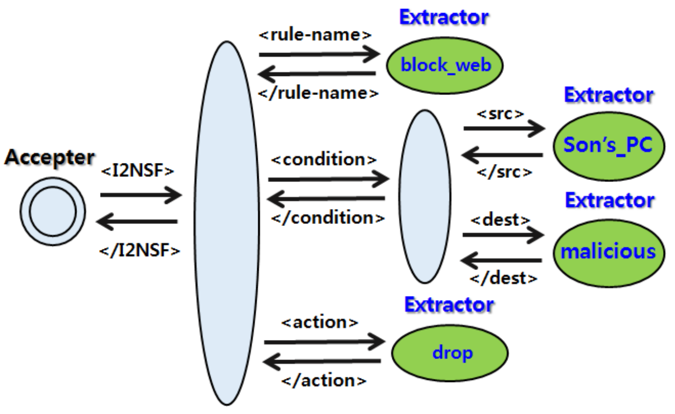


Fig. 5. Extractor based on DFA

3) Send converted data to Generator. Generator can search the proper target NSFs automatically and generate low-level policies by using converted data.

We will describe the structure of the policy translator over the next three subsections: Extractor, Data Converter, and Generator.

A. Extractor

This subsection describes Extractor based on Deterministic Finite Automaton (DFA). DFA is a finite state machine that accepts strings and produces an operation for each state transition. We used it for extracting the data from the XML tag.

Figure 5 shows architecture of Extractor. When Security Controller received the high-level policy such as Fig. 2 to Extractor, it can extract data by state transition based on XML tag. When following the state transition with the high-level policy, all of data is automatically extracted from the high-level policy.

I2NSF system manager can easily generate DFA by referring data model of Consumer-Facing Interface, because DFA structure of Extractor definitely follows the hierarchy of Consumer-Facing data model. If data model of Consumer-Facing Interface is revised, system manager need to change only DFA in Extractor for adopting to revised data model.

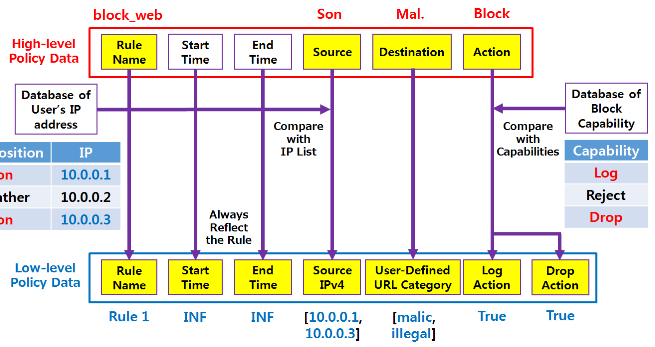


Fig. 6. Data Converter

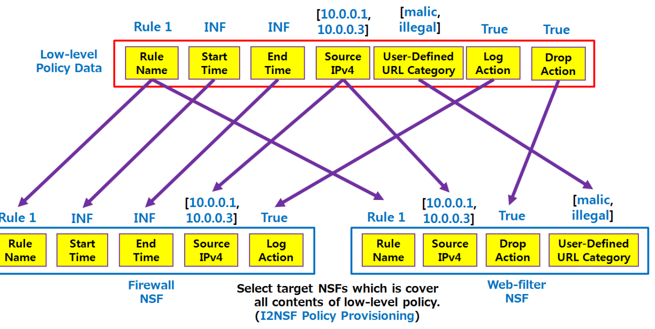


Fig. 7. Policy provisioning

B. Data Converter

This section describes Data Converter that specify data to make compatible with NSF capabilities.

If users enter the policy, which has the non-specified data, to NSF, NSF cannot recognize the data normally. For example, if the data of "son's computer" is transmitted to NSF, NSF cannot recognize the data that is not specified by IP address or the like. In order for NSF to understand the policy normally, the abstract data must be converted into the specified data which is suitable for NSF capabilities. For performing the above operation, it is necessary to compare the data with the database containing NSF capabilities, and then convert the data to the equivalent data. Figure 6 shows the process of data conversion based on database comparison. In Figure 6, the abstract data 'son' is mapped to the specific IP address '[10.0.0.1, 10.0.0.3]' by comparing with IP list in user database. By Data Converter, all of data in the high-level policy is converted to equivalent specified data which is compatible with NSF capabilities.

C. Generator

This subsection describes how to search the target NSFs automatically and how to generate low-level policy for each target NSF in Generator.

First, Generator search the proper NSFs which is able to cover all of capabilities in high-level policy. In our proposed design, Generator searches the target NSFs by comparing only NSF capabilities which is registered by Vendor's Management

System. This process is called by “policy provisioning” because Generator finds proper NSFs by using ‘only’ policy. If target NSFs are found by using other data which is not included in user’s policy, it means user already know the specific knowledge of NSF in I2NSF Framework. Figure 7 shows an example of policy provisioning. In this example, firewall NSF and web-filter NSF are selected for covering capabilities in the policy. All of capabilities can be covered by two selected NSFs.

Next, Generator makes low-level policies for each target NSF with the extracted data. We constructed Generator by using Context-free Grammar. Context-free Grammar is a set of production rules which is able to describe all possible strings in a given formal language. The low-level policy also has its own language based on YANG data model of NSF-Facing Interface. So, we can construct the productions based on YANG data model.

There are two types of production, “Content Production” and “Structure Production”. Content Production is for including data in the proper XML tag. Structure Production is for grouping other tags.

Figure 3 shows that the low-level policy consists largely of two types of tag. Tags that contain data can be created with Content Production, and tags that bundle other tags can be created with Structure Production.

Content Production can be expressed as follows:

$$[content] \rightarrow [content][content] \tag{1}$$

$$[content] \rightarrow \langle content-tag\rangle[data]\langle/content-tag\rangle \tag{2}$$

$$[data] \rightarrow data:1|data:2|...|data:n \tag{3}$$

Square brackets mean non-terminal state. If there is no square brackets, it means string is generated completely.

When duplication of tag is allowed, we can add first production (1) for rule. It is optional production. If there is no need to allow duplication, this production can be skipped.

Production (2) is the main production for Content Production. The tag which contains data can be generated by production (2). Production (3) is for injecting data in tag. If data is changed for NSF, I2NSF system manager need to change only production (3) for data mapping to each NSF.

For example, if we want to express the low-level policy for source IP address in Fig. 3, we can construct Content Production as follows:

$$[ip-content] \rightarrow [ip-content][ip-content] \tag{4}$$

$$[ip-content] \rightarrow \langle ipv4\rangle[data]\langle/ipv4\rangle \tag{5}$$

$$[data] \rightarrow 10.0.0.1|10.0.0.3 \tag{6}$$

Structure Production can be expressed as follow:

$$[struct] \rightarrow \langle start-tag\rangle[state:1]...[state:n]\langle/end-tag\rangle \tag{7}$$

$$[state:x]\;can\,be\,[struct]\;or\;[content] \tag{8}$$

Production (7) means to group other tags by the own tag name. And production (8) specifies that the various states bound to the production (7) are transferred to either the structure state or the content state.

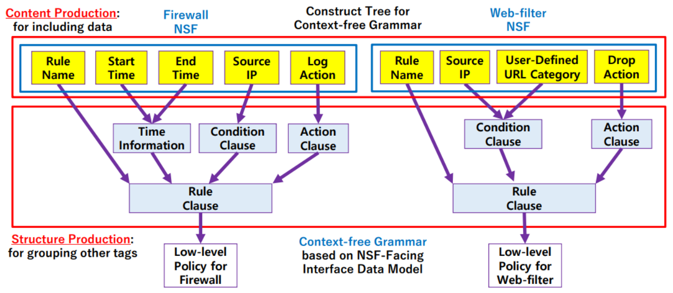


Fig. 8. Low-level policy Generator

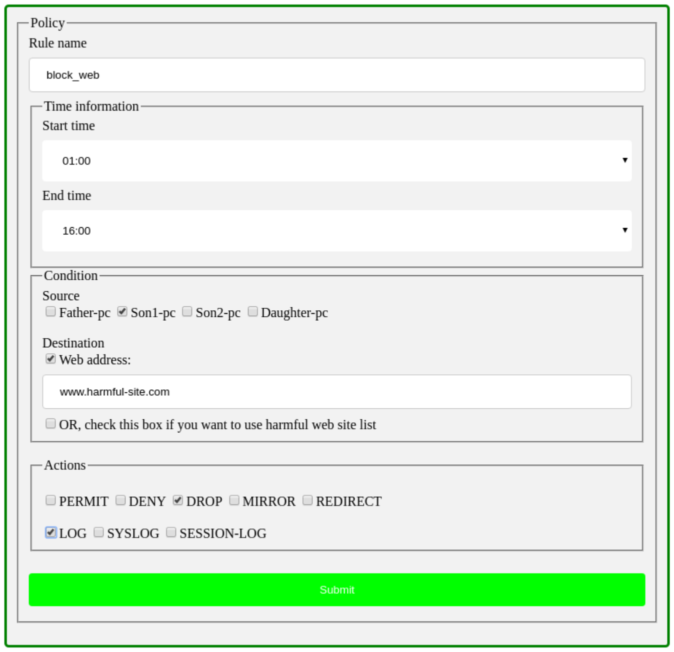

Policy
Rule name
block_web
Time information
Start time
01:00
End time
16:00
Condition
Source
☐ Father-pc ☑ Son1-pc ☐ Son2-pc ☐ Daughter-pc
Destination
☑ Web address:
www.harmful-site.com
☐ OR, check this box if you want to use harmful web site list
Actions
☐ PERMIT ☐ DENY ☑ DROP ☐ MIRROR ☐ REDIRECT
☑ LOG ☐ SYSLOG ☐ SESSION-LOG
Submit

Fig. 9. Rule creation

For example, if we want to express the low-level policy for I2NSF tag in Fig. 3, we can construct Structure Production as follows:

$$[i2nsf] \rightarrow \langle I2NSF\rangle[rule-name][rules]\langle/I2NSF\rangle \tag{9}$$

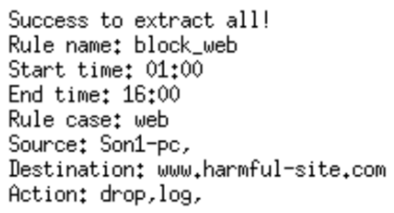
$$[rule-name]\;is\;for\,Content\,Production \tag{10}$$

$$[rules]\;is\;for\,Structure\,Production \tag{11}$$

Figure 8 shows the tree construction based on NSF-Facing Interface YANG data model. By referring the constructed tree in Fig. 8, we can easily make the productions for each clause and content. If I2NSF system manager constructs Generator based on Context-free Grammar, each target NSF can receive the generated low-level policy which contains all of required data for the target NSF.

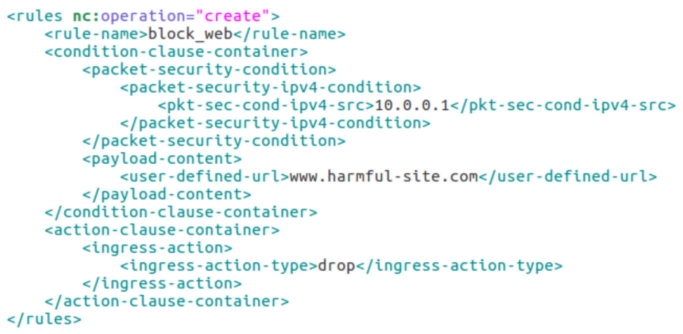
V. Implementation

In this section, we explain the implementation for proposed design of security policy translator. We constructed policy translator using Python. We received the high-level



```
Success to extract all!
Rule name: block_web
Start time: 01:00
End time: 16:00
Rule case: web
Source: Son1-pc,
Destination: www.harmful-site.com
Action: drop,log,
```

Fig. 10. Extracted data



```
<rules nc:operation="create">
    <rule-name>block_web</rule-name>
    <condition-clause-container>
        <packet-security-condition>
            <packet-security-ipv4-condition>
                <pkt-sec-cond-ipv4-src>10.0.0.1</pkt-sec-cond-ipv4-src>
            </packet-security-ipv4-condition>
        </packet-security-condition>
        <payload-content>
            <user-defined-url>www.harmful-site.com</user-defined-url>
        </payload-content>
    </condition-clause-container>
    <action-clause-container>
        <ingress-action>
            <ingress-action-type>drop</ingress-action-type>
        </ingress-action>
    </action-clause-container>
</rules>
```

Fig. 11. Low-level policy for web-filter

policy by using RESTCONF protocol, and we sent the low-level policy by using NETCONF protocol. We constructed DFA-based Extractor referred to Consumer-Facing Interface YANG data model. Also we used MySQL [9] for attaching database to Data Converter. Finally, we constructed Context-free Grammar-based Generator referred to NSF-Facing Interface YANG data model.

Figure 9 shows how to create the high-level policy. Figure 10 shows that data of the high-level policy is successfully extracted in Extractor. And then, Data Converter converted data from abstract data (e.g., Son1-pc) to specified data. (e.g., 10.0.0.1) Finally, Generator found the proper NSF (in this case, proper NSF is web-filter because of URL capability), and generate the low-level policy for the target NSF. Figure 11 shows the generated low-level policy for web-filter NSF.

VI. Conclusion

This paper proposes an Automata-based security policy translator in I2NSF framework. The proposed translator design can provide convenience to both I2NSF User and system manager. I2NSF User no longer needs to study NSF capabilities, and administrator can flexibly manage policy translation process.

For future work, we will visualize our proposed policy translator for system manager in Vendor's Management System. Then, system manager can easily debug the translation process. Also, manager can easily adopt to frequent revision of data model and user database.

Acknowledgments

This research was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (Ministry of Science and ICT, MSIT) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2018-2017-0-01633) supervised by the IITP(Institute for Information & communications Technology Promotion). Note that Jaehoon (Paul) Jeong is the corresponding author

References

[1] CIO, "The state of IT security, 2018," accessed 2018. [Online]. Available: http://www.cio.com

[2] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, and R. Kumar, "Framework for Interface to Network Security Functions," in draft-ietf-i2nsf-framework-04.pdf, IETF I2NSF WG, Feb. 2018.

[3] IETF, "The Internet Engineering Task Force," accessed 2018. [Online]. Available: https://ietf.org

[4] IETF I2NSF Working Group, "Interface to Network Security Functions(I2NSF)," accessed 2018. [Online]. Available: https://datatracker.ietf.org/wg/i2nsf/charter

[5] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF RFC 8040, Jan, 2017.

[6] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," IETF RFC 6421, Jun, 2011.

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

[8] S. Hares, R. Moskowitz, L. Xia, J. Jeong, and J. Kim, "I2NSF Capability YANG Data Model," in draft-hares-i2nsf-capability-data-model-07.pdf, IETF I2NSF WG, Mar. 2018.

[9] MySQL, "Popular Open Source Database," accessed 2018. [Online]. Available: http://www.mysql.com

[10] J. Yang, J. Jeong, and J. Kim, "Security Policy Translation in Interface to Network Security Functions," in draft-yang-i2nsf-security-policy-translation-01.pdf, IETF I2NSF WG, July 2018.