Security Policy Translation in I2NSF

draft-yang-i2nsf-security-policy-translation-02

IETF 103, Bangkok
November 7, 2018

Jinhyuk Yang, Jaehoon (Paul) Jeong, and Jinyong (Tim) Kim
Motivation for Policy Translator

• Current Situation in I2NSF
  – Different Security Policy Level Specifications exist between I2NSF User and NSFs:
    • I2NSF User: High-Level Security Policy
    • NSFs: Low-Level Security Policy

• Solution for this Situation
  – **Translation** is needed for Intent-Based Security by I2NSF User for easy security management.

• A Similar Standard (RFC 8075) from CORE WG
  – Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)
    • https://tools.ietf.org/html/rfc8075
A Previous Translation

• XSLT-based Policy Translation
  (XSLT: Extensible Stylesheet Language Transformations)
  – Proposed by W3C in 1999.

• Limitations of XSLT
  1. **Difficulty** of Security Policy Construction
     • The manager must select the proper NSF directly.
  
  2. **Inefficient Maintenance** of Data Model
     • Cannot adopt automatically the changes of a data model.
A Proposed Translation

• Automata-based Policy Translation
  – **Mapping Rules** from a High-level YANG Data Model to a Low-level YANG Data Model

• Approach
  1. **Ease** of Security Policy Construction
     • The security manager *does not need to select* a proper NSF by himself.

  2. **Efficient** Maintenance
     • Can *adopt* automatically the changes of a data model.
Updates from the Previous Version

• The Previous Drafts:
  – draft-yang-i2nsf-security-policy-translastion-01

• Changes from the previous versions
  – Add scenarios and figures for better representation of the idea.
  – The translation process is clarified with examples.
  – Other changes are described in detail in the last Appendix part.
Translation Process by Mapping

Policy Translator

I2NSF User

High-level Policy

Consumer-Facing Interface

Data Extractor

High-level Policy Data

Data Converter

NSF Database

Low-level Policy Data

Policy Generator

NSF-Facing Interface

Low-level Policy

Target NSF

High-level policy

Low-level policy

XML Example:

```xml
<I2NSF>
  <name>block_web</name>
  <cond>
    <src>Son's_PC</src>
    <dest>malicious</dest>
  </cond>
  <action>block</action>
</I2NSF>
```

```
<I2NSF>
  <rule-name>block_web</rule-name>
  <rules>
    <condition>
      <packet>
        <ipv4>10.0.0.1</ipv4>
        <ipv4>10.0.0.3</ipv4>
      </packet>
      <payload>
        <url>harm.com</url>
        <url>illegal.com</url>
      </payload>
    </condition>
    <action>drop</action>
  </rules>
</I2NSF>
```
Next Steps

• We welcome comments from WG and will modify this draft according to the comments.

• **WG Adoption Call**
  – Security Policy Translation is a core part in Security Controller.
  - This draft aims at an Informational RFC.
Appendix 1:
Process of Security Policy Translation
Step 1: Extractor (DFA)

High-level policy

Extraction

High-level policy data

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Son’s_PC</td>
</tr>
<tr>
<td>Destination</td>
<td>malicious</td>
</tr>
<tr>
<td>Action</td>
<td>block</td>
</tr>
</tbody>
</table>
Step 2: Data Converter (1/2)

High-level policy data

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Son’s_PC</td>
</tr>
<tr>
<td>Destination</td>
<td>malicious</td>
</tr>
<tr>
<td>Action</td>
<td>block</td>
</tr>
</tbody>
</table>

Low-level policy data

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IPv4</td>
<td>[10.0.0.1, 10.0.0.3]</td>
</tr>
<tr>
<td>URL Category</td>
<td>[harm.com, illegal.com]</td>
</tr>
<tr>
<td>Log Action</td>
<td>True</td>
</tr>
<tr>
<td>Drop Action</td>
<td>True</td>
</tr>
</tbody>
</table>
Step 2: Data Converter (2/2)

Low-level policy data

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IPv4</td>
<td>[10.0.0.1, 10.0.0.3]</td>
</tr>
<tr>
<td>URL Category</td>
<td>[harm.com, illegal.com]</td>
</tr>
<tr>
<td>Log Action</td>
<td>True</td>
</tr>
<tr>
<td>Drop Action</td>
<td>True</td>
</tr>
</tbody>
</table>

Log-keeper

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IPv4</td>
<td>[10.0.0.1, 10.0.0.3]</td>
</tr>
<tr>
<td>Log Action</td>
<td>True</td>
</tr>
</tbody>
</table>

Web-filter

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IPv4</td>
<td>[10.0.0.1, 10.0.0.3]</td>
</tr>
<tr>
<td>URL Category</td>
<td>[harm.com, illegal.com]</td>
</tr>
<tr>
<td>Drop Action</td>
<td>True</td>
</tr>
</tbody>
</table>
Step 3: Generator (CFG)

Low-level policy data

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>block_web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IPv4</td>
<td>[10.0.0.1, 10.0.0.3]</td>
</tr>
<tr>
<td>URL Category</td>
<td>[harm.com, illegal.com]</td>
</tr>
<tr>
<td>Drop Action</td>
<td>True</td>
</tr>
</tbody>
</table>

Generation

Low-level policy

```xml
<I2NSF>
  <rule-name>block_web</rule-name>
  <rules>
    <condition>
      <packet>
        <ipv4>10.0.0.1</ipv4>
        <ipv4>10.0.0.3</ipv4>
      </packet>
      <payload>
        <url>harm.com</url>
        <url>illegal.com</url>
      </payload>
    </condition>
    <action>drop</action>
  </rules>
</I2NSF>
```
Appendix 2: Changes from the Previous Draft
3. Necessity for Policy Translator

- Examples are added for emphasizing the necessity of translation.
- Both policies are equilibrium. The first policy is for I2NSF User, and the second policy is for NSF.
- I2NSF has a role that connects Users and NSF.
  - I2NSF requires a translator that automatically converts the first policy to the second policy even if the user gives the first one.

- Block my son's computers from malicious websites.
- Drop packets from the IP address 10.0.0.1 and 10.0.0.3 to harm.com and illegal.com
Changes from the Previous Version (2/12)

4.1. Overall Structure of Policy Translator

- NSF DB is changed to a component in the figure of an overall design of policy translator.
Changes from the Previous Versions (3/12)

4.2. DFA-based Data Extractor

- The description is clarified for better understanding.
- This Section is divided as two subsections: ‘Design’ and ‘Example Scenario’.
- The figure of DFA Architecture is changed to show the hierarchy structure.
- An example scenario and the process of the Data Extractor are added.
Changes from the Previous Version (4/12)

• 4.2. DFA-based Data Extractor
  – The figure of DFA Architecture is changed to show the hierarchy structure.
Changes from the Previous Version (5/12)

• 4.2. DFA-based Data Extractor
  – An example scenario and the process of the Data Extractor are added.
Changes from the Previous Version (6/12)

• 4.3. Data Converter
  – The description is clarified for better understanding.
  – This Section is divided as three subsections: ‘Role’, ‘Conversion’, and ‘Policy Provisioning’.
  – The role of Data Converter is emphasized.
  – The figures of data conversion and policy provisioning are added.
• 4.3. Data Converter
  – The figure and explanation of data conversion are added.
Changes from the Previous Version (8/12)

- 4.3. Data Converter
  - The figure of policy provisioning is added.

Figure 6: Example of Policy Provisioning
Changes from the Previous Version (9/12)

• 4.4. CFG-based Policy Generator
  – The description is clarified for better understanding.
  – The Section is divided as three subsections: ‘Structure Production’, ‘Content production’, and ‘Generator Construction’.
  – Examples of each production are added to help readers understand.
  – The figures for example scenario and process of the Policy Generator are added.
Changes from the Previous Version (10/12)

- 4.4. CFG-based Policy Generator
  - The examples of each production are added to help readers understand.

Example of Content Production

- `[cont_ipv4] -> [cont_ipv4][cont_ipv4]` (Allow duplication.)
- `[cont_ipv4] -> <ipv4>[cont_ipv4_data]</ipv4>
- `[cont_ipv4_data] -> 10.0.0.1 | 10.0.0.3`

Example of Structure Production

- `[struct_i2nsf] -> <I2NSF>[cont_name][struct_rules]</I2NSF>`
Changes from the Previous Version (11/12)

• 4.4. CFG-based Policy Generator
  – The figures of example scenario and the process of the Policy Generator are added.
Changes from the Previous Version (12/12)

• 6. Security Considerations
  – This Section is added. There is no security concern in policy translation.

• 8. References
  – This Section is divided by two subsections: ‘Normative References’ and ‘Informative References’.
  – References for Automata, XML(Extensible Markup Language), and XSLT(Extensible Stylesheet Language Transformations) are added.