Towards Modeling and Evaluating SPM for XML Access Control

IL-GON KIM
Korea Information Security Agency
Overview

- Motivation
- Challenging Issues
- XML Access Control Model
- Formal Security Policy Model (SPM)
  - Schema, Query, and TSP using XPath expression
  - Formal modeling of SPM using Process Algebra
- Verification Method
- Demonstration
- Conclusion
Motivation(1/2)

- Sensitive data (e.g., patient records) are increasingly becoming available in Web service paradigm and it has led to strong interest in access control to XML data.

- Access control policies for XML documents generally use path expressions such as XPath language to specify an authorized view to objects.
Motivation (2/2)

- Formal Methods is required to get high assurance levels over EAL5 in Common Criteria.

- In CC v3.1, EAL 6 requires “*Formal TOE security policy model*”.

- ADV_SPM.1
  “The model shall be in a *formal* style, supported by explanatory text as required, and identify the security policies of the TSF that are modelled. (ADV_SPM.1.1C)
Challenging Issues

- Formal modeling and verification of SPM for XML documents are non-trivial topic because of its own challenging issues:
  - the hierarchical nature in XML documents
  - the policy specification using XPath query expression which specifies the tree path
  - the TSPs which do not need to define the policy for all the nodes in XML documents; In file systems like UNIX, the TSPs are defined for every elements
XML Access Control (1/3)

Jeju hospital

Tree Structure for Patient Records

- Record
  - Patient → @Name
  - Medical
    - Doctor
    - Diagnosis
    - Prescription
XML Access Control (2/3)

Dr. Kim \[\rightarrow\] SOAP request \[\rightarrow\] Jeju hospital \[\leftrightarrow\]

$q=/\text{Record/Patient[@Name=David]}/\text{Medical/Doctor/}$
The **authorized view** is the restricted view of the XML document, which consists of the information that users are authorized to access after enforcement of the access control policies.
XPath language

- XPath is a XML path expression in order to express:
1) queries over documents, and

   ex) q = /Record/Patient[@Name=David]/Medical

2) access control rules which users are allowed or denied to access specified objects.

   ex) rule1 = <Dr. Kim, /Record/Patient[@Name=David]Medical/, read, +>
Formal Model for SPM (1/2)

- access control model for XML documents
- authorized view
- schema
- TSP
- query
- XPath
- Trace Equivalence
- CSP
- schema process
- TSP process
- query process
- formal Model
- FDR
- Model Checking
Formal Model for SPM (2/2)

- First, translate query, schema, and TSPs into automata model.
- Second, describe them formal model using process algebra language, CSP (Communicating Sequential Processes).
- Third, use FDR model checking tool.
CSP (Communicating Sequential Processes)

- The basic elements of CSP processes are action \( a \in \Sigma \) (set of actions) and they are generated by the following syntax:

\[
P ::= \text{STOP} \mid a \rightarrow P \mid P \ [\ ] P \mid P \ [\!A\!] P \mid P \mid P
\]

- STOP is deadlock termination process.
- Action prefix \( a \rightarrow P \) behaves like \( P \) when \( a \) is performed.
- \( P \ [\ ] Q \) denotes external choice.
- \( P[\!A\!] Q \) is parallel composition in which \( P \) and \( Q \) must synchronize over action in \( A \).
- \( P \mid|\mid Q \) denotes an interleaving of two processes, where each process executes entirely independently of the other until termination.
$q=/\text{Record/Patient[@Name=David]}/\text{Medical}/\ast$
Modelling $M^q$ in CSP

$Q = \text{record} \rightarrow \text{patient} \rightarrow \text{medical} \rightarrow$

(doctor $\rightarrow$ STOP [ ]

diagnosis $\rightarrow$ STOP [ ]

prescription $\rightarrow$ STOP)
Modeling TSP in CSP

- Rule1=<Dr.Kim, read, /Record/Patient/Medical/Diagnosis>

$$\text{ACR}_1 = \text{record} \rightarrow \text{patient} \rightarrow \text{medical} \rightarrow \text{diagnosis} \rightarrow \text{STOP}$$
CSP Model for XML Access Control

- Access control model AC can be described in CSP as below:
  - \( AC = S \| [A] \| TSP \)
  - \( TSP = (ACR_1 \| | ACR_2) \)
  , where A is the event of all processes.

- The AV process for the authorized view can be obtained in CSP as below:
  - \( AV = AC [ | A | ] Q \)
Verification Method (1/2)

- The query $q$ is always granted if every path accepted by the query automata $M^q$ is accepted by both the schema automata $M^s$ and access control policy automata $M^{TSP}$:
  \[ L(M^q) \subseteq L(M^s) \cap L(M^{TSP}) \]

- assert $S \parallel [A] \parallel TSF \subseteq T$ Q
Verification Methods (2/2)

\[ q = \text{Record/Patient/Medical/*} \]

\[ \text{rule 1} = /\text{Record/Patient/Medical/Diagnosis} \]
\[ \text{rule 2} = /\text{Record/Patient/Medical/Prescription} \]

*Property:* Can Dr. Kim access the sub-nodes under Medical node?

\[ L(M_q) \subseteq L(M_s) \cap (L(M_{\text{rule1}}) \cup L(M_{\text{rule2}})) \]

we can find the counterexample in CSP events:
\[ <\text{record, patient, medical, doctor}> \]

This result means that the access to the node Doctor is not permitted for Dr. Kim against the query \( q \).
Verification Result
Schema S

Record

Patient

Medical

Doctor Diagnosis Prescription
Query q and access control rules

```
q=/Record/Patient/Medical/*
```

```
Rule1=<Dr.Kim, /Record/Patient/Medical/Diagnosis, read, +>
Rule2=<Dr.Kim, /Record/Patient/Medical/Prescription, read, +>
```
Authorized View d’ of XML Document (1/2)

q=/<Record/Patient/Medical/>*

Rule1=<Dr.Kim, /Record/Patient/Medical/Diagnosis, read, +>
Rule2=<Dr.Kim, read, /Record/Patient/Medical/Prescription, read, +>

d’ = schema s \(\cap\) query q \(\cap\) rule1
Authorized View $d'$ of XML Document (2/2)

$q = /Record/Patient/Medical/*$

Rule 1 = $<$Dr.Kim, /Record/Patient/Medical/Diagnosis, read, +$>$
Rule 2 = $<$Dr.Kim, /Record/Patient/Medical/Prescription, read, +$>$

$d' = /Record/Patient/Medical//$(Diagnosis \cup \text{Prescription}$)
Conclusion

• We have presented how to specify schema, query, access control policies consisting of a SPM by interpreting XPath expression.

• We have shown how to analyze the SPM for XML documents as tree data structure in CSP language.

• Our static verification technique can not only determine whether the requested query is permitted by the schema-level access control policy or not, but also show a hierarchical path if access to data is allowed or not.