Formal Modelling of Data Integration Security Policies

Fatiuh Akeel

Supervisors: Dr. Gary B. Wills, Dr. Andrew Gravell, Dr. Federica Paci

University of Southampton- Electronics and Software Systems Group

fya1g12@soton.ac.uk

FOSAD 2016
Motivation

• Data integration systems combine data sources with different security and privacy policies to resolve queries.

• The policies contain requirements on the collection, processing, and disclosure of personal and sensitive data.

• If the security policies were not correctly enforced by the DIS → the data is exposed to data leakage threats, e.g. unauthorised disclosure or secondary use of the data.

• We aim to secure systems by correct capture and enforcement of security policies, by design.
SecureDIS helps system designers to mitigate data leakage threats during the early phases of DIS development [1].

**How?**

It provides designers with a set of *informal* guidelines:
- Based on DIS architecture
- Written in natural language
- Includes security policies
- Focuses on confidentiality, privacy, and trust to mitigate data leakage threats.
- Resulted after a conducted threat analysis[2].
Security Policies in SecureDIS
Modelling Security Policies

We apply a formal approach to model a DIS with the SecureDIS security policies and verify the correctness and consistency of the model. The model can be used as:

• a basis to perform security policies analysis
• or automatically generate a Java code to enforce those policies within DIS [3].

Event-B formal method is used for modelling

FOSAD 2016
What is Event-B?

• Extension of B Method, and is a state-based method
• Uses set theory as a main distinctive attribute
• Model systems for:
  – specification
  – verification purposes.
• Models systems gradually to reflect complexity by abstraction and refinement.
• Event-B uses mathematical proofs to ensure the correctness and consistency.

• Rodin Toolset [4]
What Does a Model in Event-B Look Like?

**CONTEXT** – The static part
- SETS
- CONSTANTS
- AXIOMS to add constraints on the sets.

**MACHINE** - The dynamic part
- VARIABLES
- INVARIANTS
- EVENTS
- VARIABLES specify the states of the system and can be modified by guarded EVENTS.
- INVARIANTS specify the constraints on variables, which need to be proved true at any state of the system.
Process of Modelling

- Policy Requirements
- Abstraction and Refinements
- Modelling
- Model Checking
Security Policies Requirements

System requirements details

<table>
<thead>
<tr>
<th>Req. no.</th>
<th>System requirement</th>
<th>Property</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Each data consumer must be assigned to a role to access data sources items</td>
<td>C</td>
<td>Specification</td>
</tr>
<tr>
<td>2</td>
<td>Each data source specifies which roles are allowed to access the sources data items</td>
<td>C</td>
<td>Specification</td>
</tr>
<tr>
<td>3</td>
<td>A data consumer is granted access to data items returned by a query if the assigned role is an allowed role</td>
<td>C</td>
<td>Enforcement</td>
</tr>
<tr>
<td>4</td>
<td>Each data consumer specifies a purpose to access data items</td>
<td>P</td>
<td>Specification</td>
</tr>
<tr>
<td>5</td>
<td>Each data item is associated with a purpose for which it was collected</td>
<td>P</td>
<td>Specification</td>
</tr>
<tr>
<td>6</td>
<td>A data consumer is granted access to data items returned by a query, if the purpose of the query matches the purpose for which the data items were collected</td>
<td>P</td>
<td>Enforcement</td>
</tr>
<tr>
<td>7</td>
<td>Each data item is classified based on its sensitivity</td>
<td>P</td>
<td>Specification</td>
</tr>
<tr>
<td>8</td>
<td>Each data consumer is assigned to a security level that specifies the authorisation to access data of a certain</td>
<td>P</td>
<td>Enforcement</td>
</tr>
</tbody>
</table>
Covered Properties

• Confidentiality
  – Access control (RBAC)

• Privacy
  – Data Use
  – Classification of data based on sensitivity

• Trust
  – Trust model
Abstraction and Refinements

Abstraction: Confidentiality -> sees -> C0

First Refinement: Privacy

refines

Second Refinement: Trust

refines

sees

extends

C1

extends

C2

FOSAD 2016
Security Policies Modelling
System Abstraction: Confidentiality

FOSAD 2016
System Abstraction: Confidentiality

- Implementing the DIS functionality
  - Create a query
  - Access data
  - Events:
    - *AddDataSources* to add data sources to the model.
    - *AddDataItemsToSources* to create data items and associate them to data sources.
    - *AddDataConsumers* to add data consumers to the model.
    - *AddRoles* to add consumers’ roles to the model.
    - *AssignRolesToConsumers* to assign consumer roles to data consumers.
    - *AddConsumersQueries* to create consumer queries containing data items.
- RBAC
  - Create an access control list
  - Enforce authorisation
System Abstraction: Confidentiality

Examples of Invariants

inv1: \( \text{belong}_\text{to} \in P_1(\text{DATA\_ITEM}) \leftrightarrow \text{sources} \)

inv2: \( \text{query} \in \text{consumers} \leftrightarrow P_1(\text{DATA\_ITEM}) \)

inv3: \( \forall c, \text{items}.c \mapsto \text{items} \in \text{query} \Rightarrow (\exists s. \text{belong}_\text{to}[\{\text{items}\}] = s) \)
Event \textit{AddAuthorisation}

\begin{align*}
\text{ANY} \quad r, i, s \\
\text{WHERE} \\
\text{grd1} : & i \in \mathbb{P}_1(\text{DATA\_ITEM}) \\
\text{grd2} : & (s \in \text{sources}) \land (\text{sources} \neq \emptyset) \\
\text{grd3} : & i \mapsto s \in \text{belong\_to} \\
\text{grd4} : & (r \in \text{roles}) \land (\text{roles} \neq \emptyset) \\
\text{grd5} : & r \mapsto i \notin \text{allowed} \\
\text{THEN} \quad \text{act1} : & \text{allowed} := \text{allowed} \cup \{r \mapsto i\} \\
\end{align*}

\text{END}
Event \textit{AccessData}
\begin{align*}
\text{ANY} \\
\text{consumer, data\_items, consumer\_roles} \\
\text{WHERE} \\
\text{grd1 : consumer} & \in \text{consumers} \\
\text{grd2 : data\_items} & \in \text{query}\{\text{consumer}\} \\
\text{grd3 : (consumer\_roles} & \subseteq \text{roles}) \land \\
& (\text{assigned}\{\text{consumer}\} = \text{consumer\_roles}) \\
\text{grd4 : } & \exists \text{role. (roles} \in \text{consumer\_roles}) \land \\
& (\text{role} \mapsto \text{data\_items} \in \text{allowed}) \\
\text{grd5 : (consumer} & \mapsto \text{data\_items}) \notin \text{access} \\
\text{THEN} \\
\text{act1 : access} & := \text{access} \cup \{\text{consumer} \mapsto \text{data\_items}\} \\
\text{END}
\end{align*}
First Refinement : Privacy

• Restricting access to queries to ensure:
  – Consumers have purposes similar to data sources purposes
    \[
    \text{axm1}: \text{partition(DATA\_USE\_PURPOSE,\{research\},\{commercial\},\{personal\},\{public\})}
    \]
  – The security clearance of the consumer, matches the sensitivity level of the data
    \[
    \text{axm2}: \text{partition(CLASSIFICATION,\{Regulated\},\{Confidential\},\{Public\})}
    \]
First Refinement : Privacy

• Access Data Event is refined to include the following guards:

\[ grd6 : \ item\_purpose[{data\_items}] = query\_purpose[{consumer}] \]

\[ grd7 : \ security\_clearance[{consumer}] = classified[{data\_items}] \]
Second Refinement : Trust

- The trust level of a consumer matches the allowed trust level of the data source.
- The trust level is calculated based on a trust model.
- The trust model calculates the consumer trust level based on risk.

By:

set \textit{TRUST\_LEVEL} containing all possible trust levels in the trust model

\begin{verbatim}
axm3:partition(TRUST\_LEVEL,{very\_good},{good},{neutral},{bad},{very\_bad})
\end{verbatim}
Second Refinement : Trust

- Access Data Event is refined to include the following guard:

\[
\text{grd8} : \text{item\_tlevel[\{data\_items\}]} = \text{consumer\_tlevel[\{consumer\}]}
\]
Evaluation of the Model

The statistics measure the Proof Obligations (PO):
• generated and discharged by the Rodin prover
• interactively proved

<table>
<thead>
<tr>
<th>Element name</th>
<th>Total</th>
<th>Auto</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>38</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>25</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>Privacy</td>
<td>9</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Trust</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

FOSAD 2016
1) Model Checking

- ProB is an animator and model checker for Event-B.
- ProB allows fully automatic exploration of Event-B models.
- Can be used to systematically check a specification for a range of errors such as deadlocks.
2) Theorem Proving

• There are different POs generated by Rodin during the development of a system

• E.g. we demonstrate an "Invariant Preservation" by ensuring that each invariant is preserved by each event.
References

[1] SecureDIS Preliminary Version  

[2] Threat Analysis on DIS 

[3] This work can be found in Data Science and Engineering Journal Article Here: 

Thank you!

Questions?