UNIVERSITY OF Southampton

Formal Modelling of Data Integration Security Policies

Fatimah 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 Framework

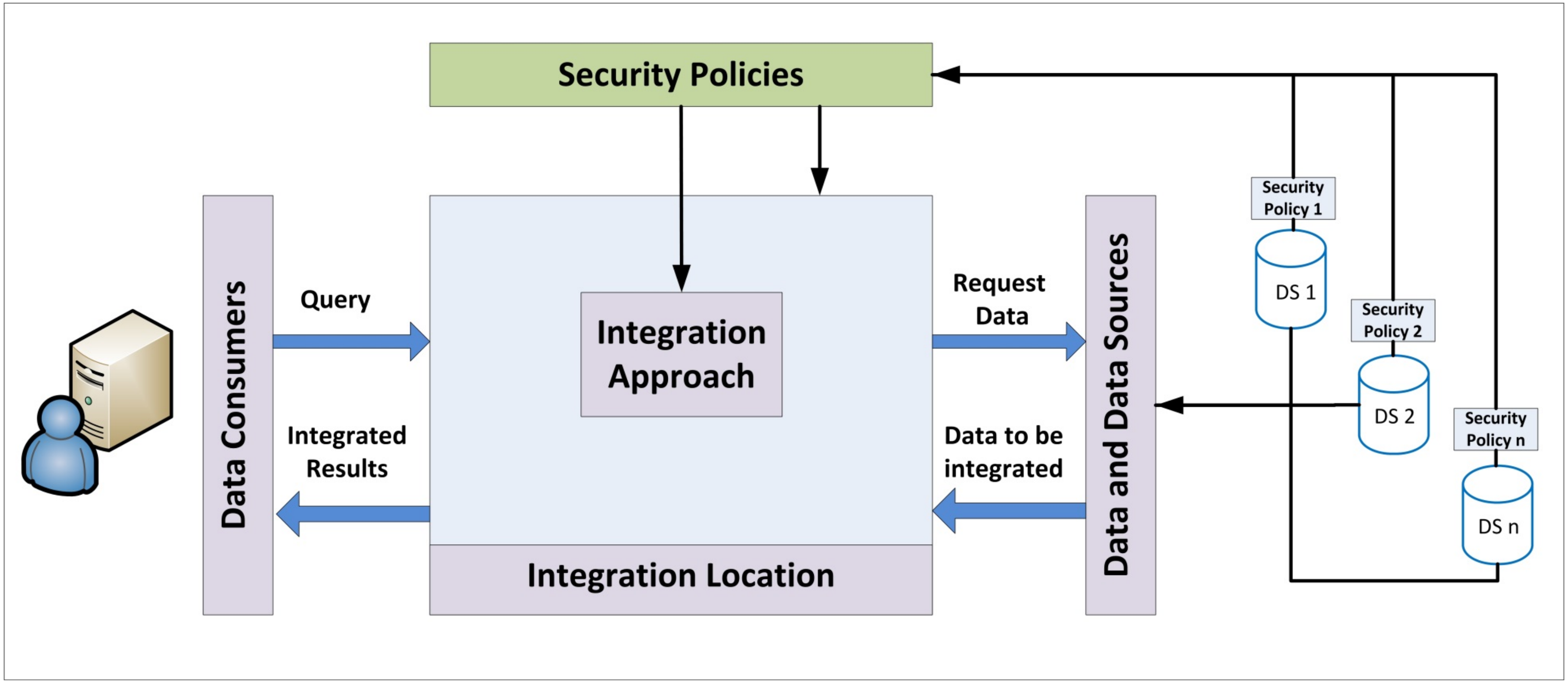
SecureDIS helps system designers to mitigate data leakage threats during the early phases of DIS development [1].

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

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?

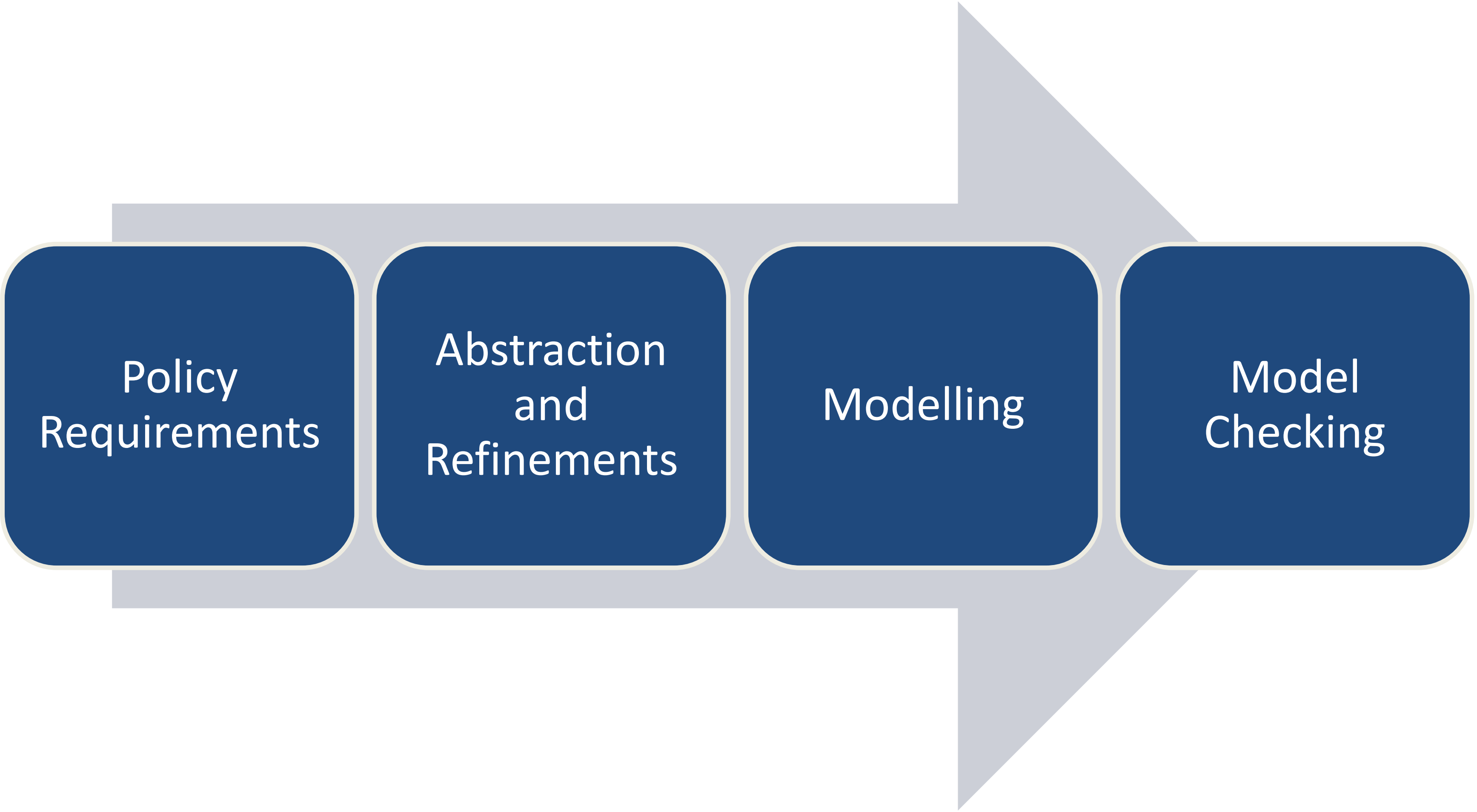
CONTEXT – The static part

- 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



Security Policies Requirements

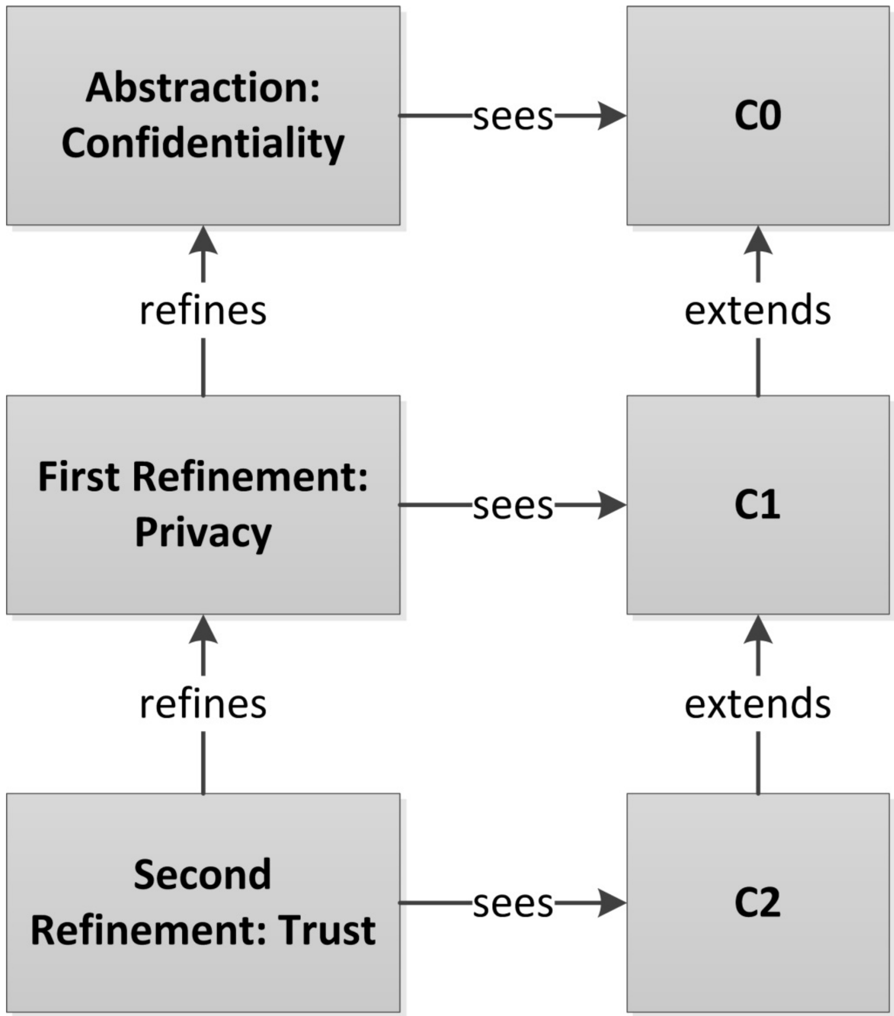
System requirements details

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

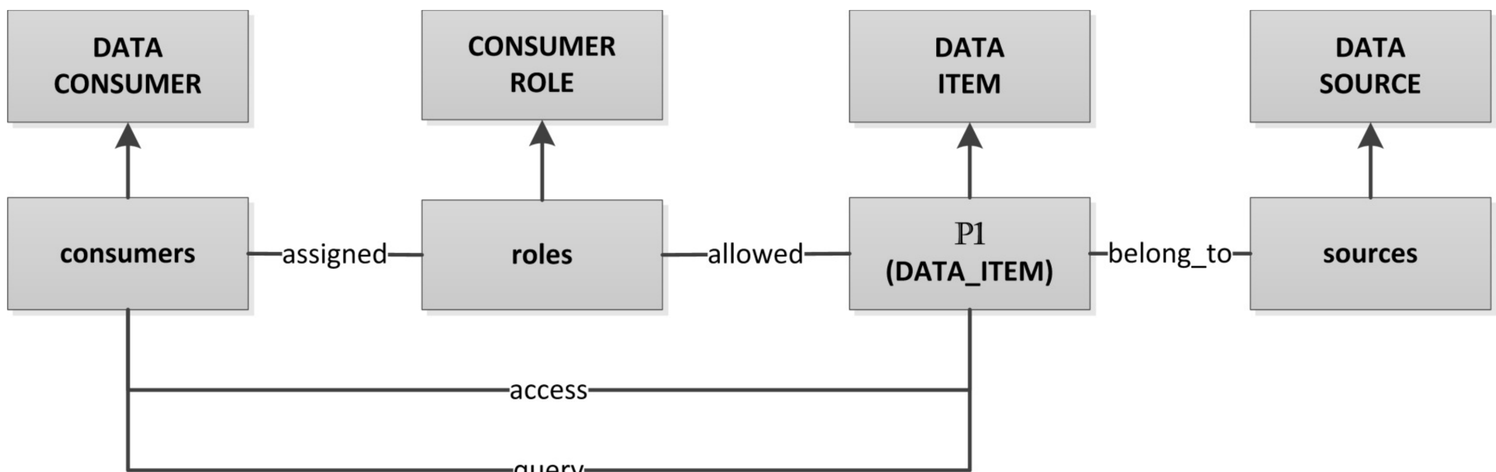
Covered Properties

- Confidentiality
  - Access control (RBAC)
- Privacy
  - Data Use
  - Classification of data based on sensitivity
- Trust
  - Trust model

Abstraction and Refinements



Security Policies Modelling
System Abstraction: Confidentiality



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’s 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

$$\textbf{inv1:}\quad belong\_to \in \mathbb{P}1(DATA\_ITEM) \leftrightarrow sources$$

$$\textbf{inv2:}\quad query \in consumers \leftrightarrow \mathbb{P}1(DATA\_ITEM)$$

$$\textbf{inv3:}\quad \forall c, items.\, c \mapsto\ items \in query \Rightarrow (\exists s.\, belong\_to[\{items\}] = s)$$

System Abstraction: Confidentiality

Event *AddAuthorisation*

ANY

r, i, s

WHERE

grd1 :$i \in \mathbb{P}1(DATA\_ITEM)$

grd2 :$(s \in sources) \wedge (sources \neq \emptyset)$

grd3 :$i \mapsto s \in belong\_to$

grd4 :$(r \in roles) \wedge (roles \notin \emptyset)$

grd5 :$r \mapsto i \notin allowed$

THEN

act1 :$allowed := allowed \cup \{r \mapsto i\}$

END

System Abstraction: Confidentiality

Event *AccessData*

ANY

$consumer, data\_items,$

$consumer\_roles$

WHERE

grd1 : $consumer \in consumers$

grd2 : $data\_items \in query[\{consumer\}]$

grd3 : $(consumer\_roles \subseteq roles) \wedge$
$(assigned[\{consumer\}] = consumer\_roles)$

grd4 : $\exists role.\,(roles \in consumer\_roles) \wedge$
$(role \mapsto data\_items \in allowed)$

grd5 : $(consumer \mapsto data\_items) \notin access$

THEN

act1 : $access := access \cup \{consumer \mapsto data\_items\}$

END

First Refinement : Privacy

- Restricting access to queries to ensure:
  - Consumers have purposes similar to data sources purposes

axm1:partition(DATA_USE_PURPOSE,{research},{commercial},{personal},{public})

  - The security clearance of the consumer, matches the sensitivity level of the data

axm2:partition(CLASSIFICATION,{Regulated},{Confidential},{Public})

First Refinement : Privacy

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

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

$$\mathbf{grd7:}\quad security\_clearance[\{consumer\}] = classified[\{data\_items\}]$$

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 *TRUST_LEVEL* containing all possible trust levels in the trust model

axm3:partition(TRUST_LEVEL,{very_good},{good},{neutral},{bad},{very_bad})

Second Refinement : Trust

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

$$\textbf{grd8 :} \quad item\_tlevel[\{data\_items\}] = consumer\_tlevel[\{consumer\}]$$

Evaluation of the Model

The statistics measure the Proof Obligations (PO):

- generated and discharged by the Rodin prover
- interactively proved

The statistics of the model

Element name	Total	Auto	Manual
Model	38	38	0
Confidentiality	25	25	0
Privacy	9	9	0
Trust	4	4	0

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 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
http://ieeexplore.ieee.org/document/6750270/

[2] Threat Analysis on DIS
http://ieeexplore.ieee.org/document/6750270/

[3] This work can be found in Data Science and Engineering Journal Article Here:
http://link.springer.com/article/10.1007/s41019-016-0016-y

[4]Event-B Info http://www.event-b.org/install.html

Thank you !

Questions?