A Framework for Secure Service Composition

Achim D. Brucker Francesco Malmignati
Madjid Merabti Qi Shi Bo Zhou

presented by
Brett Lempereur

ASE/IEEE International Conference on
Information Privacy, Security, Risk and Trust
(PASSAT)

2013-09-11

The Aniketos Project

Enable composite services to establish and maintain security and trustworthiness

Goals of the Aniketos platform:

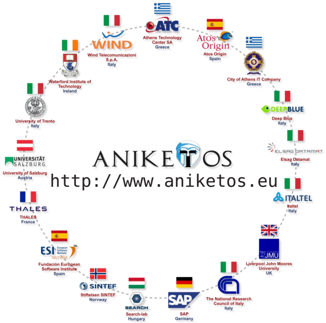
- Design-time discovery, composition and evaluation, threat awareness
- Runtime adaptation or change in service configuration
- Runtime monitoring, detection, notification

Two related dimensions:

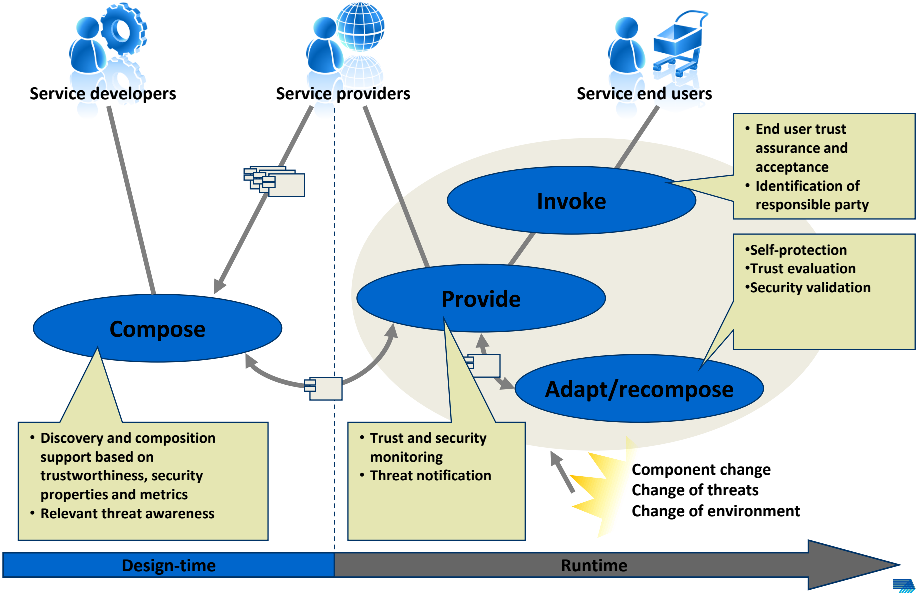
- **Trustworthiness:** Reputation, perception, centralized vs. distributed
- **Security properties:** Behavior, contracts, interfaces, formal verification

Aniketos Fact-Sheet:

- EU Integrated Project (IP), FP7 Call 5
- Budget: € 13.9 Mio (€ 9.6 Mio funding)
- 42 month (Aug. 2010 – Feb. 2014)
- Coordinator: Sintef (Norway)



The Aniketos Process



Service developers

Service providers

Service end users

Compose

Provide

Invoke

Adapt/recompose

- End user trust assurance and acceptance
- Identification of responsible party

•Self-protection
•Trust evaluation
•Security validation

- Discovery and composition support based on trustworthiness, security properties and metrics
- Relevant threat awareness

- Trust and security monitoring
- Threat notification

Component change
Change of threats
Change of environment

Design-time

Runtime

Outline

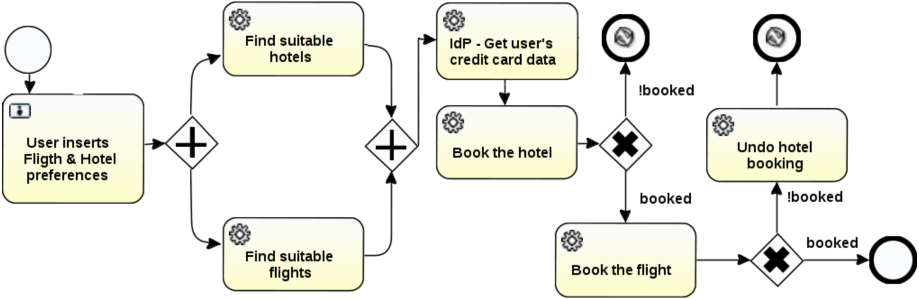
1. Motivation

2. Analysing Access Control Configurations

3. Quantifying Service Compositions

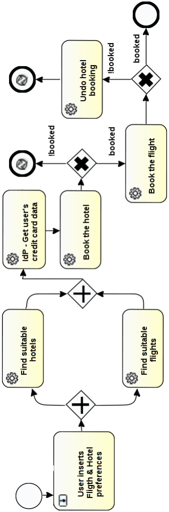
4. Conclusion

Modeling Composition Plans using BPMN



- Human-centric tasks
- Automated tasks (services)
- Orchestration of services
- Start/end states
- Logical control flow (if/and/or)
- Error states

Security and Trust Properties in Service Compositions



Access control

- Authenticated users
- Authorization of users

SoD/BoD

- No approval of own travels
- Separation of finding and booking flights

Need-to-Know

- Finding flights: only travel data
- Payment: only price and credit card data

Trust

- Use only trustworthy services
- Trustworthiness may change over time

How to ensure security, compliance, and trustworthiness at design time and runtime?

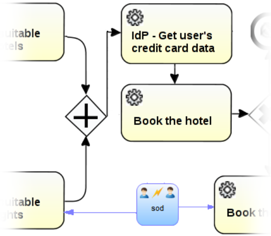
Outline

1. Motivation
2. Analysing Access Control Configurations
3. Quantifying Service Compositions
4. Conclusion

The Problem: RBAC with Separation of Duty

Role-based access control (RBAC)

- Subjects are assigned to roles
- Permissions assign roles to tasks (resources)



Separation of duty (SoD)

- restrict subjects in executing tasks

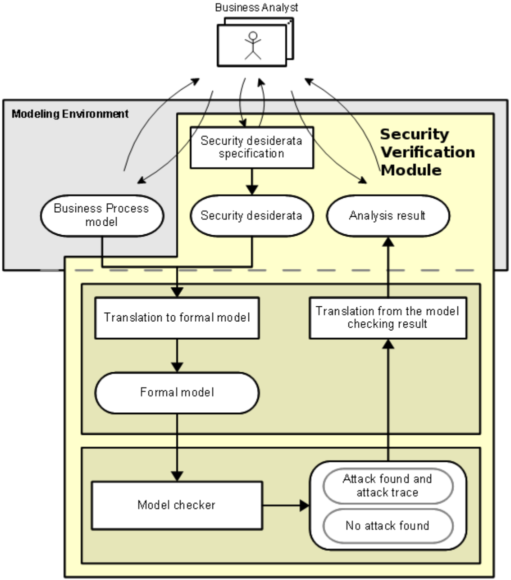
We analyze:

- Does the RBAC configuration comply to the SoD requirements?

  yes: static SoD
  no: dynamic SoD

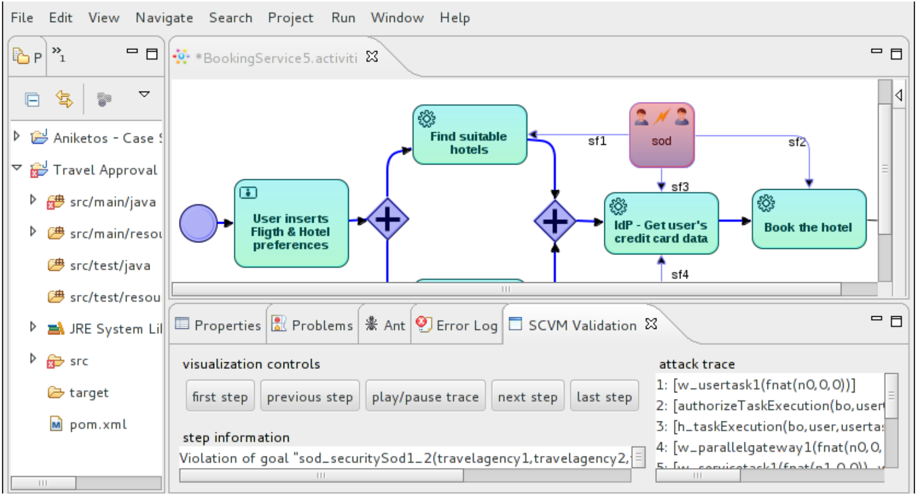
- In case of a compliance violation:
  - change RBAC configuration
  - ensure dynamic enforcement of SoD

Security Verification Module (RBAC/SoD Check)



Business Analyst
Modeling Environment
Security Verification Module
Security desiderata specification
Business Process model
Security desiderata
Analysis result
Translation to formal model
Translation from the model checking result
Formal model
Model checker
Attack found and attack trace
No attack found

User Interface for the Service Designer



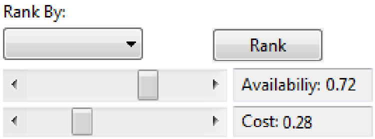
Violation of goal "sod_securitySod1_2(travelagency1,travelagency2,

```
1: [w_usertask1(fnat(n0,0,0))]
2: [authorizeTaskExecution(bo,user
3: [h_taskExecution(bo,user,userta
4: [w_parallelgateway1(fnat(n0,0,
5: [w_servicetask1(fnat(n1,0,0)),
```

Outline

1. Motivation
2. Analysing Access Control Configurations
3. Quantifying Service Compositions
4. Conclusion

The Problem: Selection of the Optimal Composition



Availabiliy: 0.72

Cost: 0.28

- Ranking of service compositions
  - property of the composition
  - compositions provide the same
    - functionality
    - security and trustworthiness
- Ranking according to
  - Availability
  - Costs

Ranking Secure Service Compositions

- Calculating the availability:

	Description	Calculation
○→○	Sequence	$\prod_{i=1}^{n} A_i$
+	Parallel	$\min(A_1, \ldots, A_n)$
×	Exclusive	A_i

- Calculating the costs:

$$C = \sum_{i=1}^{n} C_i$$

Example: Ranking Service Compositions

- Assume the following availability values:
  - Find suitable hotels: 0.99
  - Find suitable flights: 0.96
  - Get user's credit card data: 0.97
  - Book the hotel: 0.99
  - Book the flight: 0.98
  - Undo hotel booking: 0.94
- We compute:

$$A = \min(0.99, 0.96) \times 0.97 \times 0.99 \times 0.98 = 0.90$$

- Assume the weights to 0.72 (availability) and 0.28 (cost)

$$V = 0.72 \times A + 0.28 \times \frac{B - C}{B}$$

Outline

1 Motivation

2 Analysing Access Control Configurations

3 Quantifying Service Compositions

4 Conclusion

Conclusion and Outlook

- Secure service compositions require:
  - **Design time:**
    modeling, analysis and ranking of secure services
  - **Run-time:**
    enforcement, monitoring, service replacement, and re-planning

- Today, we presented design time support for
  - Analysing security properties of service compositions
  - a method for ranking service compositions

- Our work is part of the Aniketos secure Composition Framework
- Further information about Aniketos: `http://www.aniketos.eu`

Thank you for your attention!

Any questions or remarks?

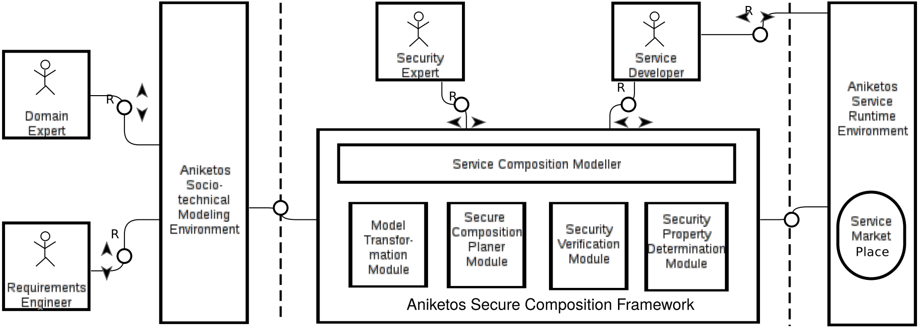
Further Readings

Achim D. Brucker, Francesco Malmignati, Madjid Merabti, Qi Shi, and Bo Zhou.
A framework for secure service composition.
In *ASE/IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*. IEEE Computer Society, 2013.

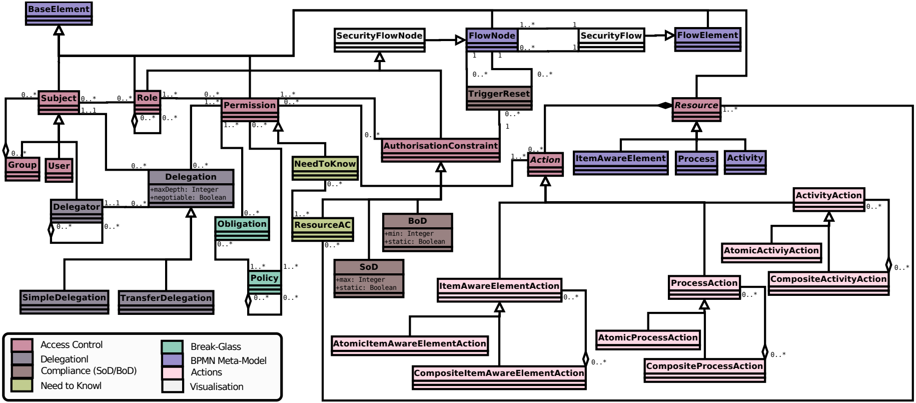
Part II

Appendix

The Aniketos Secure Composition Framework


Domain Expert
Requirements Engineer
R
R
Aniketos Socio-technical Modeling Environment
Security Expert
R
Service Developer
R
R
Service Composition Modeller
Model Transfor-mation Module
Secure Composition Planer Module
Security Verification Module
Security Property Determination Module
Aniketos Secure Composition Framework
Aniketos Service Runtime Environment
Service Market Place

SecureBPMN: Adding Security Specifications



- Access Control
- Delegation
- Separation/Binding of Duty
- Need to Know
- Break Glass

Analyzing (Dynamic | Static) Separation of Duty

Does the access control enforce a separation of duty constraint

- Translate the composition plan to ASLan

```
hc rbac_ac(Subject, Role, Task) := CanDoAction(Subject, Role, Task)
           :- user_to_role(Subject, Role), poto(Role, Task)
hc poto_T6 := poto(Staff, Request Travel)
hc poto_T6 := poto(Manager, Approve Absence)
hc poto_T7 := poto(Manager, Approve Budget)
```

- Specify the test goal

```
attack_state sod_securitySod1_1(Subject0,Subject1,Instance1,Instance2)
   := executed(Subject0,task(Request Travel,Instance1)).
      executed(Subject1,task(Approve Budget,Instance2)).
      executed(Subject3,task(Approve Absence,Instance3))
      &not(equal(Subject0,Subject1))
      &not(equal(Subject1,Subject2))
      &not(equal(Subject2,Subject3))
```

- Run the model checker
- Translate the analysis result back to BPMN (visualization)