Information Flow Security for Asynchronous, Distributed, and Mobile Applications

Felipe LUNA DEL AGUILA

OASIS Project
INRIA Sophia Antipolis
CNRS - I3S - Université Nice–Sophia Antipolis
Agenda

• Introduction
Agenda

- Introduction

- Context (*informal* and *formal* perspectives)
Agenda

• Introduction

• Context (*informal* and *formal* perspectives)
  – ProActive
Agenda

• Introduction

• Context (*informal* and *formal* perspectives)
  – ProActive
  – *ASP* calculus and communication reduction rules
Agenda

- Introduction

- Context (*informal* and *formal* perspectives)
  - ProActive
  - ASP calculus and communication reduction rules

- Objectives
Agenda

- Introduction

- Context (informal and formal perspectives)
  - ProActive
  - ASP calculus and communication reduction rules

- Objectives

- Related Security Mechanisms
Agenda

• Introduction

• Context (*informal* and *formal* perspectives)
  – ProActive
  – ASP calculus and communication reduction rules

• Objectives

• Related Security Mechanisms

• The ASP Security Model
Agenda

• Introduction

• Context (*informal* and *formal* perspectives)
  – ProActive
  – ASP calculus and communication reduction rules

• Objectives

• Related Security Mechanisms

• The ASP Security Model

• Implementation of the Security Model
Agenda

- Introduction

- Context (*informal* and *formal* perspectives)
  - ProActive
  - ASP calculus and communication reduction rules

- Objectives

- Related Security Mechanisms

- The ASP Security Model

- Implementation of the Security Model

- Conclusion
Introduction

Recent paradigms
Introduction

Recent paradigms

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Introduction

Recent paradigms

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Introduction

Recent paradigms

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Introduction

Recent paradigms

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Distributed Systems
Service-oriented Computing
Object-oriented Programming
Security
Introduction

Recent paradigms

Security focused specifically on *Information Flow*
Context

ProActive main characteristics

• Middleware library for distributed applications
Context

ProActive main characteristics

• Middleware library for distributed applications

• 100% Java
Context

ProActive main characteristics

• Middleware library for distributed applications

• 100% Java

• Existence of passive and active objects
Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and active objects
- Asynchronous communications between active objects
Context

ProActive main characteristics

• Middleware library for distributed applications

• 100% Java

• Existence of passive and active objects

• Asynchronous communications between active objects
  – Principle of wait-by-necessity and futures:
    1. future reference
Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects
  - Principle of *wait-by-necessity* and *futures*:
    1. future reference
       (ex.: http://www.anysite.com/anypage.html)
Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and active objects
- Asynchronous communications between active objects
  - Principle of wait-by-necessity and futures:
    1. future reference
       (ex.: http://www.anysite.com/anypage.html)
    2. future value
Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects
  - Principle of *wait-by-necessity and futures*:
    1. future reference
       (ex.: http://www.anysite.com/anypage.html)
    2. future value
       (ex.: HTML error: 404 Not Authorized)
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects, activity $\beta$
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects, activity $\beta$, active object reference $AO(\alpha)$
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects, activity $\beta$, active object reference $AO(\alpha)$, future $f_{i}^{\alpha \rightarrow \beta}$ and request queue (with pending, current, and completed requests)
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects, activity $\beta$, active object reference $AO(\alpha)$, future $f_{i}^{\alpha \rightarrow \beta}$ and request queue (with pending, current, and completed requests), future references $fut(f_{i}^{\alpha \rightarrow \beta})$
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects, activity $\beta$, active object reference $AO(\alpha)$, future $f_{i}^{\alpha \rightarrow \beta}$ and request queue (with pending, current, and completed requests), future references $fut(f_{i}^{\alpha \rightarrow \beta})$, and store $\sigma$
ASP semantics and ProActive

- The ASP language entities:
  - activity $\alpha$, active object $a$, passive objects, activity $\beta$, active object reference $AO(\alpha)$, future $f_{i}^{\alpha\rightarrow\beta}$ and request queue (with pending, current, and completed requests), future references $fut(f_{i}^{\alpha\rightarrow\beta})$, and store $\sigma$

- Parallel configurations are then of the form: $P,Q ::= \alpha [a; \sigma; \upsilon; F; R; f] \| \beta [\cdots] \| \cdots$
ASP parallel reduction rules
ASP parallel reduction rules

(LOCAL)

\[(a, \sigma) \rightarrow_{\mathcal{S}} (a', \sigma') \rightarrow_{\mathcal{S}} \text{does not clone a future}\]

\[\alpha[a; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[a'; \sigma'; \iota; F; R; f] \parallel P\]

(NEWACT)

\[\gamma \text{ fresh activity} \quad \iota' \not\in \text{dom}(\sigma) \quad \sigma' = \{\iota' \mapsto \text{AO}(\gamma)\} :: \sigma \quad \sigma_\gamma = \text{copy}(\iota'', \sigma)\]

\[\alpha[\mathcal{R}[\text{Active}(\iota'', m_j)]; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[\mathcal{R}[\iota']; \sigma'; \iota; F; R; f] \parallel \gamma[\iota''.m_j(); \sigma_\gamma; \iota''; \emptyset; \emptyset; \emptyset] \parallel P\]

(REQUEST)

\[\sigma_\alpha(\iota) = \text{AO}(\beta) \quad \iota'' \not\in \text{dom}(\sigma_\beta) \quad f_i^{\alpha \rightarrow \beta} \text{ new future} \quad \iota_f \not\in \text{dom}(\sigma_\alpha) \quad \sigma'_\beta = \text{CopyMerge}(\sigma_\alpha, \iota'; \sigma_\beta, \iota'') \quad \sigma'_\alpha = \{\iota_f \mapsto \text{fut}(f_i^{\alpha \rightarrow \beta})\} :: \sigma_\alpha\]

\[\alpha[\mathcal{R}[\iota.m_j(\iota')]]; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow \]

\[\alpha[\mathcal{R}[\iota_f]]; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \iota_\beta; F_\beta; R_\beta; [m_j; \iota''; f_i^{\alpha \rightarrow \beta}]; f_\beta] \parallel P\]

(SERVE)

\[R = R' :: [m_j; \iota_r; f'] :: R'' \quad m_j \in M \quad \forall m \in M, m \notin R' \]

\[\alpha[\mathcal{R}[\text{Serve}(M)]; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[\iota.m_j(\iota_r) \uparrow f, \mathcal{R}[]]; \sigma; \iota; F; R'; R''; f'] \parallel P\]

(ENDSERVICE)

\[\iota' \not\in \text{dom}(\sigma) \quad F' = F :: \{f \mapsto \iota'\} \quad \sigma' = \text{CopyMerge}(\sigma, \iota'; \sigma, \iota')\]

\[\alpha[\iota \uparrow f', a; \sigma; \iota; F; R; f] \parallel P \rightarrow \alpha[a; \sigma'; \iota; F'; R'; f'] \parallel P\]

(REPLY)

\[\sigma_\alpha(\iota) = \text{fut}(f_i^{\gamma \rightarrow \beta}) \quad F_\beta(f_i^{\gamma \rightarrow \beta}) = \iota_f \quad \sigma'_\alpha = \text{CopyMerge}(\sigma_\beta, \iota_f; \sigma_\alpha, \iota)\]

\[\alpha[a_\alpha; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow \]

\[\alpha[a_\alpha; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P\]
ASP parallel reduction rules

\[(a, \sigma) \rightarrow_{S} (a', \sigma') \quad \rightarrow_{S} \text{does not clone a future} \quad \text{(LOCAL)}\]

\[\alpha[a; \sigma; \nu; F; R; f] \parallel P \rightarrow \alpha[a'; \sigma'; \nu; F; R; f] \parallel P\]

\[\gamma \text{ fresh activity} \quad \nu' \notin \text{dom}(\sigma) \quad \sigma' = \{\nu' \mapsto \text{AO}(\gamma)\} :: \sigma \quad \sigma_\gamma = \text{copy}(\nu'', \sigma) \quad \text{(NEWACT)}\]

\[\alpha[\text{R[Active}\left(\nu'', m_j\right)]; \sigma; \nu; F; R; f] \parallel P \rightarrow \alpha[\text{R[\nu']}; \sigma'; \nu; F; R; f] \parallel \gamma[\nu''.m_j()]; \sigma_\gamma; \nu''; \emptyset; \emptyset; \emptyset] \parallel P\]

\[\sigma_\alpha(\nu) = \text{AO}(\beta) \quad \nu'' \notin \text{dom}(\sigma_\beta) \quad f_i^{\alpha \rightarrow \beta} \text{ new future} \quad \nu_f \notin \text{dom}(\sigma_\alpha) \quad \sigma'_\alpha = \text{Copy&Merge}(\sigma_\alpha, \nu'; \sigma_\beta, \nu'') \quad \sigma_\alpha' = \{\nu_f \mapsto \text{fut}(f_i^{\alpha \rightarrow \beta})\} :: \sigma_\alpha \quad \text{(REQUEST)}\]

\[\alpha[\text{R[\nu.m_j(\nu')]}]; \sigma_\alpha; \nu_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \nu_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow P\]

\[\alpha[\text{R[\nu_f]}]; \sigma'_\alpha; \nu_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \nu_\beta; F_\beta; R_\beta; [m_j; \nu''; f_i^{\alpha \rightarrow \beta}]; f_\beta] \parallel P\]

\[R = R' :: [m_j; \nu_r; f'] :: R'' \quad m_j \in M \quad \forall m \in M, m \notin R' \quad \text{(SERVE)}\]

\[\alpha[\text{R[Serve}(M)); \sigma; \nu; F; R; f] \parallel P \rightarrow \alpha[\nu.m_j(\nu_r) \uparrow f, \text{R[]}]; \sigma; \nu; F; R' :: R'''; f'] \parallel P\]

\[\nu' \notin \text{dom}(\sigma) \quad F' = F :: \{f \mapsto \nu'\} \quad \sigma' = \text{Copy&Merge}(\sigma, \nu; \sigma, \nu') \quad \text{(ENDSERVICE)}\]

\[\alpha[\nu \uparrow f', a; \sigma; \nu; F; R; f] \parallel P \rightarrow \alpha[a; \sigma'; \nu; F'; R; f'] \parallel P\]

\[\sigma_\alpha(\nu) = \text{fut}(f_i^{\gamma \rightarrow \beta}) \quad F_\beta(f_i^{\gamma \rightarrow \beta}) = \nu_f \quad \sigma_\alpha' = \text{Copy&Merge}(\sigma_\beta, \nu_f; \sigma_\alpha, \nu) \quad \text{(REPLY)}\]

\[\alpha[a_\alpha; \sigma_\alpha; \nu_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \nu_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow P\]

\[\alpha[a_\alpha; \sigma'_\alpha; \nu_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \nu_\beta; F_\beta; R_\beta; f_\beta] \parallel P\]
Reduction rules: 1) New Activity

\( \gamma \) fresh activity
Reduction rules: 1) New Activity

\( \gamma \) fresh activity \quad \text{copy}(i'', \sigma)
Reduction rules: 1) New Activity

\[ \gamma \text{ fresh activity} \quad \text{copy}(i'', \sigma) = \sigma_\gamma \]
Reduction rules: 1) New Activity

\[ \gamma \text{ fresh activity} \quad \text{copy}(i'', \sigma) = \sigma_{\gamma} \quad i' \notin \text{dom}(\sigma) \quad \sigma' = \{i' \mapsto AO(\gamma)\} :: \sigma \]

\[ \alpha[\mathcal{R}[\text{Active}(i'', m_j)];\sigma; i; F; R; f] \parallel P \rightarrow \alpha[\mathcal{R}[i'];\sigma'; i; F; R; f] \parallel \gamma[i''.m_j(); \sigma_{\gamma}; i''; \emptyset; \emptyset; \emptyset] \parallel P \]
Reduction rules: 2) Request

\[ \sigma_\alpha(i) = AO(\beta) \]
Reduction rules: 2) Request

\[ \sigma_\alpha(\iota) = AO(\beta) \quad \iota'' \notin \text{dom}(\sigma_\beta) \quad \text{Copy\&Merge}(\sigma_\alpha, \iota'; \sigma_\beta, \iota'') \]
Reduction rules: 2) Request

\[ \sigma_\alpha(i) = AO(\beta) \quad i'' \notin \text{dom}(\sigma_\beta) \quad \text{Copy\&Merge}(\sigma_\alpha, i' ; \sigma_\beta, i'') = \sigma'_\beta \]
Reduction rules: 2) Request

\[ \sigma_\alpha(i) = AO(\beta) \quad i'' \notin \text{dom}(\sigma_\beta) \quad \text{Copy\&Merge}(\sigma_\alpha, i' ; \sigma_\beta, i'') = \sigma'_\beta \]

\[ f^{\alpha \rightarrow \beta}_i \quad \text{new future} \]
Reduction rules: 2) Request

\[ \sigma_\alpha(i) = AO(\beta) \quad i'' \notin \text{dom}(\sigma_\beta) \quad \text{Copy&Merge}(\sigma_\alpha, i'; \sigma_\beta, i'') = \sigma'_\beta \]

\[ f_{i}^{\alpha \rightarrow \beta} \quad \text{new future} \quad i_f \notin \text{dom}(\sigma_\alpha) \quad \{ i_f \mapsto \text{fut}(f_{i}^{\alpha \rightarrow \beta}) \} :: \sigma_\alpha = \sigma'_\alpha \]
Reduction rules: 2) Request

\[
\sigma_\alpha(i) = AO(\beta) \quad i'' \notin \text{dom}(\sigma_\beta) \quad \text{Copy&Merge}(\sigma_\alpha, i' ; \sigma_\beta, i'') = \sigma'_\beta \\
\text{new future} \quad \text{new future} \quad \{ \xi_f \mapsto \text{fut}(f_{i \mapsto \beta}) \} :: \sigma_\alpha = \sigma'_\alpha
\]

\[
\alpha[\mathcal{R}[i.m_j(i')]; \sigma_\alpha; i_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \xi_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow \\
\alpha[\mathcal{R}[i_f]; \sigma'_\alpha; i_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \xi_\beta; F_\beta; R_\beta :: [m_j; i'']; f_{i \mapsto \beta}'; f_\beta] \parallel P
\]
Reduction rules: 3) Reply

\[ \sigma_\alpha(\iota) = \text{fut} (f^\gamma_i) \quad F_\beta(f^\gamma_i) = \iota_f \]
Reduction rules: 3) Reply

\[ \sigma_\alpha(\nu) = \text{fut}(f_i^\gamma \rightarrow^\beta) \quad F_\beta(f_i^\gamma \rightarrow^\beta) = \nu_f \quad \text{Copy}&\text{Merge}(\sigma_\beta, \nu_f ; \sigma_\alpha, \nu) \]
Reduction rules: 3) Reply

\[
\sigma_\alpha(\nu) = fut(f_\gamma^{\alpha \rightarrow \beta}) \quad F_\beta(f_\gamma^{\alpha \rightarrow \beta}) = \nu_f \quad Copy&Merge(\sigma_\beta, \nu_f ; \sigma_\alpha, \nu) = \sigma'_\alpha
\]

\[
\alpha[a_\alpha; \sigma_\alpha; \nu_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \nu_\beta; F_\beta; R_\beta; f_\beta] \parallel P \rightarrow \\
\alpha[a_\alpha; \sigma'_\alpha; \nu_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \nu_\beta; F_\beta; R_\beta; f_\beta] \parallel P
\]
Objectives

Main objective: Information Flow Control
Objectives

Main objective: Information Flow Control

1. To guarantee data confidentiality

Confidentiality in MLS: follows the basic principle of no write down, no read up
Objectives

Main objective: Information Flow Control

1. To guarantee data confidentiality

Confidentiality in MLS: follows the basic principle of no write down, no read up

2. Define a security policy to apply to asynchronous, distributed, and mobile applications
Main objective: Information Flow Control

1. To guarantee data confidentiality

Confidentiality in MLS: follows the basic principle of no write down, no read up

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

Main issue: Presence of asymmetric patterns of communications (result of the future and wait-by necessity concepts)
Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   **Confidentiality in MLS:** follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

   **Main issue:** Presence of asymmetric patterns of communications (result of the *future and wait-by necessity* concepts)

3. Provide a formal security model
Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   **Confidentiality in MLS:** follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

   **Main issue:** Presence of asymmetric patterns of communications (result of the *future* and *wait-by necessity* concepts)

3. Provide a formal security model *which is verifiable mathematically*
Objectives

Main objective: Information Flow Control

1. To guarantee data confidentiality
   
   Confidentiality in MLS: follows the basic principle of no write down, no read up

2. Define a security policy to apply to asynchronous, distributed, and mobile applications
   
   Main issue: Presence of asymmetric patterns of communications (result of the future and wait-by necessity concepts)

3. Provide a formal security model which is verifiable mathematically

4. Propose an architecture for the implementation of the security model
Related Security Mechanisms

Agenda

• Introduction
• Context
• Objectives
• **Mechanisms**
• ASP Security Model
• Implementation
• Conclusions

• **Models**
Related Security Mechanisms

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

● Models
  - Mandatory Access Controls (MAC)
Related Security Mechanisms

• Models

  – Mandatory Access Controls (MAC)
  – Discretionary Access Controls (DAC)
Related Security Mechanisms

**Models**

- Mandatory Access Controls (MAC)
- Discretionary Access Controls (DAC)

**Formal methods**
Related Security Mechanisms

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

• Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

• Formal methods
  - Non-interference
Related Security Mechanisms

• Models
  – Mandatory Access Controls (MAC)
  – Discretionary Access Controls (DAC)

• Formal methods
  – Non-interference
  – $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

- Formal methods
  - Non-interference
  - $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  - Ambients, and Mobile Ambients
Related Security Mechanisms

• Models
  – Mandatory Access Controls (MAC)
  – Discretionary Access Controls (DAC)

• Formal methods
  – Non-interference
  – $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  – Ambients, and Mobile Ambients
  – Seals, Boxed Ambients, and Secure Safe Ambients
Related Security Mechanisms

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

• Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

• Formal methods
  - Non-interference
  - $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients

• Informal approaches
Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

- Formal methods
  - Non-interference
  - $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients

- Informal approaches
  - Exceptions (in the security policy)
Related Security Mechanisms

- **Models**
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

- **Formal methods**
  - Non-interference
  - $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients

- **Informal approaches**
  - Exceptions (in the security policy), labels
Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

- Formal methods
  - Non-interference
  - $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients

- Informal approaches
  - Exceptions (in the security policy), labels, tickets
Related Security Mechanisms

• Models
  – Mandatory Access Controls (MAC)
  – Discretionary Access Controls (DAC)

• Formal methods
  – Non-interference
  – $\pi$-calculus, Asynchronous-$\pi$, and Spi calculus
  – Ambients, and Mobile Ambients
  – Seals, Boxed Ambients, and Secure Safe Ambients

• Informal approaches
  – Exceptions (in the security policy), labels, tickets, certificates . . .
The ASP Security Model

Syntax and semantics of the security framework
The ASP Security Model

Syntax and semantics of the security framework

- $\mathcal{S}$ set of activities acting as subjects, where $\alpha, \beta, \gamma, \ldots \in \mathcal{S}$
The ASP Security Model

Syntax and semantics of the security framework

• $S$ set of activities acting as subjects, where $\alpha, \beta, \gamma, ... \in S$

• $D$ set of data objects sent as arguments in REQUEST actions: $Rq_{\alpha \rightarrow \beta}(d)$
The ASP Security Model

Syntax and semantics of the security framework

- $S$ set of activities acting as subjects, where $\alpha, \beta, \gamma, \ldots \in S$

- $D$ set of data objects sent as arguments in REQUEST actions: $Rq_{\alpha \rightarrow \beta}(d)$

- $R$ is the set of objects associated to futures, and returned in REPLY actions: $Rp_{\beta \rightarrow \alpha}(r)$
The ASP Security Model (cntd.)

- $\mathcal{L}$ finite set of security levels $\lambda$, partially ordered by the relation $\leq$, where $\forall i \in S \cup D, \lambda_i \in \mathcal{L}$
The ASP Security Model (cntd.)

- $\mathcal{L}$ finite set of security levels $\lambda$, partially ordered by the relation $\leq$, where $\forall i \in S \cup D, \lambda_i \in \mathcal{L}$.

---

**Agenda**

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
The ASP Security Model (cntd.)

- $\mathcal{L}$ finite set of security levels $\lambda$, partially ordered by the relation $\leq$, where $\forall i \in S \cup D, \lambda_i \in \mathcal{L}$

- Transmissions of $d$ and $r$ are restricted by the security rules
The ASP Security Model (cntd.)

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
The ASP Security Model \textit{(cntd.)}

**Actions**

- \( \mathcal{A} \) set of actions, where \( a \in \mathcal{A} \)

- ASP actions are now rewritten to include security properties:
The ASP Security Model (cntd.)

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
- ASP actions are now rewritten to include security properties:
  $$NW(\gamma, \lambda_{\gamma})$$ is a modified NEWACT
The ASP Security Model (cntd.)

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$

- ASP actions are now rewritten to include security properties:

\[ \text{NEWACT}(\gamma, \lambda_\gamma) \] is a modified NEWACT

\[ \text{REQUEST}_\alpha \rightarrow_\beta(d, \lambda_{in}) \] is a modified REQUEST
The ASP Security Model (cntd.)

Actions

• $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$

• ASP actions are now rewritten to include security properties:

\[ Nw(\gamma, \lambda_\gamma) \] is a modified NEWACT
\[ Rq_{\alpha \rightarrow \beta}(d, \lambda_{in}) \] is a modified REQUEST
\[ Rp_{\beta \rightarrow \alpha}(r) \] is an unchanged REPLY
The ASP Security Model (cntd.)

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$

- ASP actions are now rewritten to include security properties:

  $\text{Nw}(\gamma, \lambda_\gamma)$ is a modified NEWACT
  $\text{Rq}_{\alpha \rightarrow \beta}(d, \lambda_{in})$ is a modified REQUEST
  $\text{Rp}_{\beta \rightarrow \alpha}(r)$ is an unchanged REPLY

- In general,
  
  $a = \{\text{Nw}(\gamma, \lambda_\gamma), \text{Rq}_{\alpha \rightarrow \beta}(d, \lambda_{in}), \text{Rp}_{\beta \rightarrow \alpha}(r)\}$
The ASP Security Model (cntd.)

Additional entities

• \( M \) matrix of explicit (discretionary) rights
The ASP Security Model (cntd.)

Additional entities

- $M$ matrix of explicit (discretionary) rights

\[ M = S \times S \rightarrow \mathcal{P}(A) \]
The ASP Security Model (cntd.)

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

$$\mathcal{M} = S \times S \rightarrow \mathcal{P}(\mathcal{A})$$

- $\mathcal{P}(\mathcal{A})$ set of actions whose assignation of a security level is explicitly allowed
The ASP Security Model (cntd.)

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

$$\mathcal{M} = S \times S \rightarrow \mathcal{P}(A)$$

$\mathcal{P}(A)$ set of actions whose assignation of a security level is explicitly allowed

In general, $p \in \mathcal{P}(A)$ if and only if

$$p = \{Nw(\gamma, \lambda_\gamma), Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})\}$$
The ASP Security Model (cntd.)

Additional entities

• $\mathcal{M}$ matrix of explicit (discretionary) rights

$$\mathcal{M} = \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$$

$\mathcal{P}(\mathcal{A})$ set of actions whose assignment of a security level is explicitly allowed

In general, $p \in \mathcal{P}(\mathcal{A})$ if and only if

$$p = \{Nw(\gamma, \lambda_\gamma), Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})\}$$

• $\mathcal{T}$ set of authorized (access) transmissions
The ASP Security Model (cntd.)

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights
  \[
  \mathcal{M} = S \times S \rightarrow \mathcal{P}(A)
  \]
  $\mathcal{P}(A)$ set of actions whose assignation of a security level is explicitly allowed
  In general, $p \in \mathcal{P}(A)$ if and only if
  \[
  p = \{NW(\gamma, \lambda_\gamma), Rq_\alpha\rightarrow_\beta(d, \lambda_{in})\}
  \]

- $\mathcal{T}$ set of authorized (access) transmissions
  \[
  \mathcal{T} = S \times S \times A
  \]
Application of the security framework

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- **ASP Security Model**
- Implementation
- Conclusions
Application of the security framework

1.- Base statements:

- **Creation (and migration) of new activities are secure**
- **Emission of requests, with modifiable security levels in the data sent, are secure**
- **Emission of replies are secure**
Application of the security framework

1.- Base statements:
   • Creation (and migration) of new activities are secure
   • Emission of requests, with modifiable security levels in the data sent, are secure
   • Emission of replies are secure

2.- Support concepts:
   • Elementary flows of information
   • Flow-paths
Application of the security framework

1.- Base statements:

- Creation (and migration) of new activities are secure
- Emission of requests, with modifiable security levels in the data sent, are secure
- Emission of replies are secure

2.- Support concepts:

- Elementary flows of information
- *Flow-paths*

3.- Results:

Confidentiality, from end-to-end in a flow-path, is guaranteed
Secure Activity Creation

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Secure Activity Creation

\[ \forall \alpha, \gamma \in S \]
Secure Activity Creation

\[ \forall \alpha, \gamma \in S: (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in T \iff \]

A new activity action is considered secure iff:
Secure Activity Creation

\[ \forall \alpha, \gamma \in S: (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in T \iff (\lambda_\alpha \leq \lambda_\gamma) \]

A new activity action is considered secure iff:

1. The new activity has a higher security level compared to its creator
Secure Activity Creation

\( \forall \alpha, \gamma \in S: (\alpha, \gamma, NW(\gamma, \lambda_\gamma)) \in T \iff (\lambda_\alpha \leq \lambda_\gamma) \lor NW(\gamma, \lambda_\gamma) \in M(\alpha, \gamma) \)

A new activity action is considered secure iff:

1. The new activity has a higher security level compared to its creator

2. or, in case the new activity has a lower security (i.e. a downgrade), the creation action must be explicitly allowed
Secure Activity Creation

\[ \forall \alpha, \gamma \in S: (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in T \iff (\lambda_\alpha \leq \lambda_\gamma) \lor Nw(\gamma, \lambda_\gamma) \in M(\alpha, \gamma) \]

A new activity action is considered secure iff:

1. The new activity has a higher security level compared to its creator

2. or, in case the new activity has a lower security (i.e. a downgrade), the creation action must be explicitly allowed

Special case: Migration of an existing activity
Secure Request Transmission

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Secure Request Transmission

\[ \forall \alpha, \beta \in S \]

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Secure Request Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, Rq_{\alpha\rightarrow\beta}(d, \lambda_{in})) \in T \iff \]

A request transmission action is considered secure iff:
Secure Request Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})) \in T \iff (\lambda_{in} \leq \lambda_{\beta}) \land \]

A request transmission action is considered secure iff:

1. Data is ”released” to an authorized target, AND
Secure Request Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})) \in T \iff (\lambda_{in} \leq \lambda_{\beta}) \land (\lambda_{\alpha} \leq \lambda_{in}) \]

A request transmission action is considered secure iff:

1. Data is ”released” to an authorized target, \textit{AND}

2. Either:
   - The data has a higher level than the sender
Secure Request Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})) \in T \iff (\lambda_{in} \leq \lambda_{\beta}) \land \left( (\lambda_{\alpha} \leq \lambda_{in}) \lor ((\lambda_{\alpha} > \lambda_{in}) \land Rq_{\alpha \rightarrow \beta}(d, \lambda_{in}) \in M(\alpha, \beta)) \right) \]

A request transmission action is considered secure iff:
1. Data is ”released” to an authorized target, AND
2. Either:
   - The data has a higher level than the sender
   - If data has a lower level than the sender (i.e. a downgrade), the action must be explicitly allowed
Secure Request Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})) \in T \iff \lambda_{in} \leq \lambda_{\beta} \land \left( (\lambda_{\alpha} \leq \lambda_{in}) \lor ((\lambda_{\alpha} > \lambda_{in}) \land Rq_{\alpha \rightarrow \beta}(d, \lambda_{in}) \in M(\alpha, \beta)) \lor \exists \gamma, \delta, f_i, d = fut(f_i^{\gamma \rightarrow \delta}) \right) \]

A request transmission action is considered secure iff:

1. Data is "released" to an authorized target, AND

2. Either:
   - The data has a higher level than the sender
   - If data has a lower level than the sender (i.e. a downgrade), the action must be explicitly allowed
   - The data is a future reference
Secure Reply Transmission

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Secure Reply Transmission

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

\[ \forall \alpha, \beta \in S \]
Secure Reply Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, R_{\beta \rightarrow \alpha}(r)) \in T \iff \]

A reply transmission action is considered secure iff:

---

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Secure Reply Transmission

∀α, β ∈ S: (α, β, Rpβ→α(r)) ∈ T ⇐⇒

(λβ ≤ λα)

A reply transmission action is considered secure iff:

1. The data contained in the reply r (hence of level λβ) can be released to the corresponding receiving subject (with λα)
Secure Reply Transmission

\[ \forall \alpha, \beta \in S: (\alpha, \beta, Rp_{\beta \rightarrow \alpha}(r)) \in T \iff (\lambda_\beta \leq \lambda_\alpha) \lor (\exists \gamma, \delta, f_i, r = fut(f_{i}^{\gamma \rightarrow \delta})) \]

A reply transmission action is considered secure iff:

1. The data contained in the reply \( r \) (hence of level \( \lambda_\beta \)) can be released to the corresponding receiving subject (with \( \lambda_\alpha \))

2. or, if the data in the reply is only a reference to a future
Secure ASP reduction rules
Secure ASP reduction rules

\[ \gamma \text{ fresh activity} \quad I' \not\in \text{dom}(\sigma) \quad \sigma' = \{ I' \mapsto AO(\gamma) \} :: \sigma \]

\[ \sigma_\gamma = \text{copy}(I'', \sigma) \quad (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in T \]

(SecureNEWACT)

\[ \alpha^\lambda[\mathcal{R}[\text{Active}^\lambda(\alpha(I''), m_j)]; \sigma; I; F; R; f] \parallel P \rightarrow \alpha^\lambda[\mathcal{R}[I']; \sigma'; I; F; R; f] \parallel \gamma^\lambda[I''.m_j()]; \sigma_\gamma; I''; \emptyset; \emptyset; \emptyset] \parallel P \]

\[ \sigma_\alpha(I) = AO(\beta) \quad I'' \not\in \text{dom}(\sigma_\beta) \quad f_{i}^{\alpha\rightarrow\beta} \text{ new future} \]

\[ \sigma'_\beta = \text{Copy&Merge}(\sigma_\alpha, I'; \sigma_\beta, I'') \]

\[ \sigma'_\alpha = \{ I_f \mapsto \text{fut}(f_{i}^{\alpha\rightarrow\beta}) \} :: \sigma_\alpha \quad (\alpha, \beta, Rq_{\alpha\rightarrow\beta}(\sigma_\alpha(I'), \lambda_{in})) \in T \]

(SecureREQUEST)

\[ \alpha^{\lambda\alpha}[\mathcal{R}[I.m_j(I'_{\lambda\text{in}})]; \sigma_\alpha; I_{\alpha}; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta^{\lambda\beta}[a_{\beta}; \sigma_\beta; I_{\beta}; F_{\beta}; R_{\beta}; f_{\beta}] \parallel P \rightarrow \alpha^{\lambda\alpha}[\mathcal{R}[I_f]; \sigma'_\alpha; I_{\alpha}; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta^{\lambda\beta}[a_{\beta}; \sigma'_\beta; I_{\beta}; F_{\beta}; R_{\beta}; :: [m_j; I''; f_{i}^{\alpha\rightarrow\beta}; f_{\beta}]] \parallel P \]

\[ \sigma_\alpha(I) = \text{fut}(f_{i}^{\gamma\rightarrow\beta}) \quad F_{\beta}(f_{i}^{\gamma\rightarrow\beta}) = I_f \]

\[ \sigma'_\beta = \text{Copy&Merge}(\sigma_\alpha, I_f; \sigma_\beta, I) \quad (\beta, \alpha, R_{p_{\beta\rightarrow\alpha}}(\sigma_\beta(I_f))) \in T \]

(SecureREPLY)

\[ \alpha^{\lambda\alpha}[a_{\alpha}; \sigma_\alpha; I_{\alpha}; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta^{\lambda\beta}[a_{\beta}; \sigma_\beta; I_{\beta}; F_{\beta}; R_{\beta}; f_{\beta}] \parallel P \rightarrow \alpha^{\lambda\alpha}[a_{\alpha}; \sigma'_\alpha; I_{\alpha}; F_{\alpha}; R_{\alpha}; f_{\alpha}] \parallel \beta^{\lambda\beta}[a_{\beta}; \sigma_\beta; I_{\beta}; F_{\beta}; R_{\beta}; f_{\beta}] \parallel P \]
Secure ASP reduction rules

γ fresh activity \quad i' \notin \text{dom}(\sigma) \quad \sigma' = \{i' \mapsto AO(\gamma)\} :: \sigma

\sigma_\gamma = \text{copy}(i''', \sigma) \quad (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in T

(\text{SecNEWACT})

\alpha^\lambda[\mathcal{R}[\text{Active}^\lambda(\sigma', m_j)]; \sigma; i; F; R; f] \parallel P \rightarrow
\alpha^\lambda[\mathcal{R}[\sigma']; \sigma'; i; F; R; f] \parallel \gamma^\lambda[a''(i'''. \sigma_\gamma); i'''; \emptyset; \emptyset; \emptyset] \parallel P

(\sigma_\alpha(\iota) = AO(\beta) \quad i'' \notin \text{dom}(\sigma_\beta) \quad f_\beta^\lambda \rightarrow^\gamma \text{ new future}
\iota_f \notin \text{dom}(\sigma_\alpha) \quad \sigma_\beta' = \text{Copy} \& \text{Merge}(\sigma_\alpha, \iota'; \sigma_\beta, i''')

\sigma_\alpha' = \{i_f \mapsto \text{fut}(f_\beta^\lambda \rightarrow^\gamma)\} :: \sigma_\alpha \quad (\alpha, \beta, Rq_{\alpha \rightarrow \beta}(\sigma_\alpha(\iota'), \lambda_{\text{in}})) \in T

(\text{SecREQUEST})

\alpha^\lambda[\mathcal{R}[\iota.m_j(i'\lambda_{\text{in}})]; \sigma_\gamma; \iota; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^\lambda[\beta_{\alpha \rightarrow \beta}(a_\beta; \sigma_\gamma; \iota; F_\beta; R_\beta; f_\beta)] \parallel P \rightarrow
\alpha^\lambda[\mathcal{R}[\iota_f]; \sigma_\alpha'; \iota; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^\lambda[\beta_{\alpha \rightarrow \beta}(a_\beta; \sigma_\alpha'; \iota; F_\beta; R_\beta; f_\beta) :: [m_j; i'''; f_\beta^\lambda \rightarrow^\gamma]; f_\beta] \parallel P

\sigma_\alpha(\iota) = \text{fut}(f_\gamma \rightarrow^\beta)
\sigma_\alpha' = \text{Copy} \& \text{Merge}(\sigma_\beta, \iota_f; \sigma_\alpha, \iota)
\beta_\alpha(\sigma_\alpha(\iota_f)) \in T

(\text{SecREPLY})

\beta^\lambda[\beta_{\alpha \rightarrow \beta}[a_\beta; \sigma_\alpha; \iota; F_\alpha; R_\alpha; f_\alpha] \parallel P \rightarrow
\beta^\lambda[\beta_{\alpha \rightarrow \beta}[a_\beta; \sigma_\alpha'; \iota; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^\lambda[\beta_{\alpha \rightarrow \beta}[a_\beta; \sigma_\alpha; \iota; F_\beta; R_\beta; f_\beta] \parallel P

Parallel configurations are now of the form:
P, Q ::= \alpha^\lambda[a; \sigma; \iota; F; R; f] \parallel \beta^\lambda[\cdots] \parallel \cdots
The Secure Information Flow

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
The Secure Information Flow

- The concept *elementary flow of information* is based on the "release" or transmission of information from an activity.
The Secure Information Flow

- The concept *elementary flow of information* is based on the "release" or transmission of information from an activity

- Hence, it is derived the secure information flow notion:

\[
\begin{align*}
(\alpha, \beta, Rq_{\alpha \rightarrow \beta}(\sigma(\nu'), \lambda_{in})) \in T & \quad \Rightarrow \quad Sec\varphi_{0}(\alpha, \beta) \\
(\beta, \alpha, Rp_{\beta \rightarrow \alpha}(\sigma_{\alpha}(\nu_{f}))) \in T & \quad \Rightarrow \quad Sec\varphi_{0}(\beta, \alpha) \\
(\alpha, \gamma, Nw(\gamma, \lambda_{\gamma})) \in T & \quad \Rightarrow \quad Sec\varphi_{0}(\alpha, \gamma)
\end{align*}
\]
The Secure Information Flow

- The concept *elementary flow of information* is based on the "release" or transmission of information from an activity.

- Hence, it is derived the *secure information flow* notion:

\[
(\alpha, \beta, Rq_{\alpha \rightarrow \beta}(\sigma(\iota'), \lambda_{in})) \in T \\
Sec\varphi_0(\alpha, \beta) \\

(\beta, \alpha, Rp_{\beta \rightarrow \alpha}(\sigma_{\alpha}(\iota_f))) \in T \\
Sec\varphi_0(\beta, \alpha) \\

(\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in T \\
Sec\varphi_0(\alpha, \gamma)
\]

The syntax \( Sec\varphi_0(\alpha, \beta) \) means there is a secure flow \( (Sec\varphi) \), with no other intermediate activities \( (\emptyset) \), happening between activities \( \alpha \) and \( \beta \)
The Secure Path for Information Flow

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
The Secure Path for Information Flow

A flow of information is composed of several elementary flows happening in a sequential order
The Secure Path for Information Flow

**Agenda**
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

- A *flow of information* is composed of several elementary flows happening in a sequential order

- A *flow-path* ($fp$) is produced when intermediate activities are present in between the communication of two given activities (i.e. the end points)
The Secure Path for Information Flow

- A *flow of information* is composed of several elementary flows happening in a sequential order.

- A *flow-path* \((fp)\) is produced when intermediate activities are present in between the communication of two given activities (i.e. the end points).

- Formally, the *secure path for information flow* is:

$$\frac{\text{Sec}_\varphi_{fp_1}(\alpha, \gamma) \quad \text{Sec}_\varphi_{fp_2}(\gamma, \beta)}{\text{Sec}_\varphi_{fp_1 \cdot \gamma \cdot fp_2}(\alpha, \beta)}$$
A flow of information is composed of several elementary flows happening in a sequential order.

A flow-path \((fp)\) is produced when intermediate activities are present in between the communication of two given activities (i.e. the end points).

Formally, the secure path for information flow is:

\[
\frac{\text{Sec}_\varphi_{fp_1}(\alpha, \gamma) \quad \text{Sec}_\varphi_{fp_2}(\gamma, \beta)}{\text{Sec}_\varphi_{fp_1.\gamma.fp_2}(\alpha, \beta)}
\]

There is a secure information flow from end-to-end on any flow path when:

\[
\text{Sec}_\varphi_{\gamma_1...\gamma_n}(\alpha, \beta) \iff \text{Sec}_\varphi_\emptyset(\alpha, \gamma_1) \land \text{Sec}_\varphi_\emptyset(\gamma_1, \gamma_2) \land \cdots \land \text{Sec}_\varphi_\emptyset(\gamma_n, \beta)
\]
Service-Oriented Computing and futures

Impossible future updates with symmetric patterns of communications

\[ \lambda_\delta \leq \lambda_\beta < \lambda_\gamma \]
Service-Oriented Computing and *futures*

**Agenda**
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

**Impossible future updates with symmetric patterns of communications**

\[ \lambda_\delta \leq \lambda_\beta < \lambda_\gamma \]
Service-Oriented Computing and *futures*

Impossible future updates with symmetric patterns of communications

\[ \lambda_\delta \leq \lambda_\beta < \lambda_\gamma \]
Impossible future updates with symmetric patterns of communications

\[ \beta \leq \lambda \beta < \lambda \gamma \]
Impossible future updates with symmetric patterns of communications

\[ \lambda_\delta \leq \lambda_\beta < \lambda_\gamma \]
Impossible future updates with symmetric patterns of communications

\[ \beta \xrightarrow{f_2} AO(\gamma) \xrightarrow{Req_\gamma} AO(\delta) \xrightarrow{f_2^\prime} \delta \]

\[ \lambda_\delta \leq \lambda_\beta < \lambda_\gamma \]

\[ \lambda_\delta \leq \lambda_\gamma \checkmark \]
Service-Oriented Computing and *futures*

Impossible future updates with symmetric patterns of communications

Diagram with labeled elements and relations: AO(γ), AO(δ), f_2, Req_γ, Req_δ.
Service-Oriented Computing and *futures* (contd.)

Future updates are possible in asymmetric patterns of communications

![Diagram showing asymmetric patterns of communications]
Future updates are possible in asymmetric patterns of communications
Future updates are possible in asymmetric patterns of communications

\[ \lambda_\delta \leq \lambda_\beta < \lambda_\gamma \]

\[ \lambda_\delta \leq \lambda_\beta \checkmark \]
Implementation of the Security Model

Architecture of active objects

**Agenda**
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

![Diagram of active objects architecture](image)
Implementation of the Security Model (contd.)

Security schema for *active objects*

---

**Agenda**
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

---

**Application layer**

A

**ProActive middleware layer**

Stub-B

**Body**

B
Implementation of the Security Model (contd.)

Security schema for *active objects*

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Implementation of the Security Model (contd.)

Security schema for *active objects*

**Agenda**
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Implementation of the Security Model (contd.)

Security schema for active objects

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Diagram:
- Application layer
  - A
  - B
  - Request
  - Stub-B
  - Security sublayer
    - EF / DF
    - AF
  - ProActive middleware layer
  - Java layer
    - JVM X
    - JVM Y
    - Java API
Implementation of the Security Model (contd.)

Security schema for active objects

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Implementation of the Security Model (contd.)

Security schema for active objects

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Implementation of the Security Model (contd.)

Security schema for active objects

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Diagram:

- Application layer
- ProActive middleware layer
- Security sublayer
- Java layer

Diagram components:

- JVM X
- JVM Y
- Java API
- EF / DF
- AF
- Stub-B
- Body
- A
- B
- Request
- Reply
- Authorized request
- Authorized reply

Diagram legend:

- EF / DF
- AF
- Stub-B
- Body
- JVM
- Java API
Implementation of the Security Model (contd.)

Detailed Security sub-layer
Implementation of the Security Model (contd.)

Detailed Security sub-layer

- EF = flow control mechanism as a Java Security Manager

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

intercepted action
(newActive, turnActive, request, reply, or migrateTo)
Implementation of the Security Model \textit{(contd.)}

**Agenda**
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

**Detailed Security sub-layer**

- \textbf{EF} = flow control mechanism as a Java Security Manager
- \textbf{DF} = Context Handler + Policy Decision Point + XACML file

\textbf{Diagram:}
- **DF**
  - PDP
  - XACML policy file
  - ContextHandler
  - setup
  - intercepted action
    - (newActive, turnActive, request, reply, or migrateTo)

\textbf{Java API:}
- flow control Security Manager
Implementation of the Security Model (contd.)

Detailed Security sub-layer

- **EF = flow control mechanism as a Java Security Manager**
- **DF = Context Handler + Policy Decision Point + XACML file**
- **AF = Policy Information Point + active object PIP**

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

EF = flow control Security Manager

intercepted action
(newActive, turnActive, request, reply, or migrateTo)

DF

PDP

XACML policy file

setup

ContextHandler

AF

PIP

AOPIP

Java API
Implementation of the Security Model (contd.)

Detailed Security sub-layer

- EF = flow control mechanism as a Java Security Manager
- DF = Context Handler + Policy Decision Point + XACML file
- AF = Policy Information Point + active object PIP

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

intercepted action
(newActive, turnActive, request, reply, or migrateTo)

\[DF\]

- PDP
- XACML policy file
- setup

\[AF\]

- ContextHandler
- setup

\[EF\]

- flow control Security Manager
- Java API

\[PIP\]

\[AOPIP\]
Conclusions

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions
Conclusions

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

- **Expresiveness:**
  - Assignation of specific security levels to request parameters and created activities
Conclusions

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

• **Expresiveness:**
  - Assignation of specific security levels to request parameters and created activities

• **Scalability:**
  - Dynamic checks performed only at activity creation, and inter-activity communications
Conclusions

Agenda

• Introduction
• Context
• Objectives
• Mechanisms
• ASP Security Model
• Implementation
• Conclusions

• Expresiveness:
  – Assignation of specific security levels to request parameters and created activities

• Scalability:
  – Dynamic checks performed only at activity creation, and inter-activity communications

• Extendable:
  – XACML features provide a finer control on the discretionary access control
Perspectives

Agenda

• Introduction
• Context
• Objectives
• Mechanisms
• ASP Security Model
• Implementation
• Conclusions
Perspectives

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)
Perspectives

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)

- Further study of covert channels in distributed systems
Perspectives

TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)

Further study of covert channels in distributed systems

Static type checking in Java can be complemented with our model
Perspectives

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)
- Further study of covert channels in distributed systems
- Static type checking in Java can be complemented with our model
- The security mechanism can be applied to the Components paradigm
Q&A

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Questions?

Thank you for your attention
Q&A

Agenda
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Questions ?

Thank you for your attention