A Quality of Context-Aware Approach to Access Control in Pervasive Environments

Alessandra Toninelli
Antonio Corradi Rebecca Montanari

Università di Bologna
Department of Electronics, Information and Systems

The Second International ICST Conference on
Mobile Wireless Middleware, Operating Systems, and Applications
Berlin, 28 April 2009

Outline

Outline

Context-Aware Access Control Policies

Example

In case of emergency, any qualified physician located within the hospital is allowed to access Alice's health protected information.

- Access control policies
  - High level directives defining *who* can access *which resource* under *which circumstances*
- Traditional policies based on identities/roles – static
- **Context-aware** access control policies
  - Definition of policies based on context
  - Use of semantic technologies to represent & reason about policies and contexts

Why We Need Quality of Context

- Context information determines behavior & strategies of context-aware security systems
- But is context information *reliable* itself?
  - Unknown – no available sensor information
  - Ambiguous – conflicting information from different sources
  - Imprecise – information with insufficient granularity
  - Erroneous – sensed or aggregated context not coherent with real situation
- Need to explicitly consider the **quality of context** used in security decisions

QoC Awareness & Security – State of the Art

1. QoC representation
   - Declarative models (QoC metadata)
   - Logical models (e.g., FOL, fuzzy logics)
   - Probabilistic models
2. QoC calculation techniques
   - Learning techniques and sensor training (e.g., to calculate precision & correctness)
   - Algorithms to calculate QoC value of aggregated contexts
   - Algorithms to deal with uncertain/ambiguous contexts (conflict resolution)
3. Few context-aware architectures explicitly consider QoC
   - Impact of QoC on context-aware security has not been thoroughly analyzed yet

Outline

Proteus Context & Policy Model

- Proteus policies associate resources to *protection contexts*
- A protection context is a set of attributes & constrained values (*context elements*)
- The current state is a set of attributes & values measured by "sensors" (*context assertions*)
  - A protection context is *active* if context assertions describing the current state match its context elements
- Activation of protection contexts (and associated policies) allows access to a resource

Context & Policy Representation

Example

In case of emergency, any qualified physician located within the hospital is allowed to access Alice's health protected information.

$$\begin{aligned}&\text{PersonalEmergencyContext} \equiv \\ &\text{ProtectionContext} \sqcap \exists\ \text{owner.Alice} \sqcap \exists\ \text{requestor.InHospitalQualifiedPhysician} \sqcap \\ &\exists\ \text{resource.AliceHPI} \sqcap \exists\ \text{environment.PersonalEmergency}\end{aligned}$$

$$< Dr.Green, located, EmergencyRoom >$$
$$< CurrentState, environment, PersonalEmergency >$$

- Context & policy representation based on Description Logic

Quality of Context Model

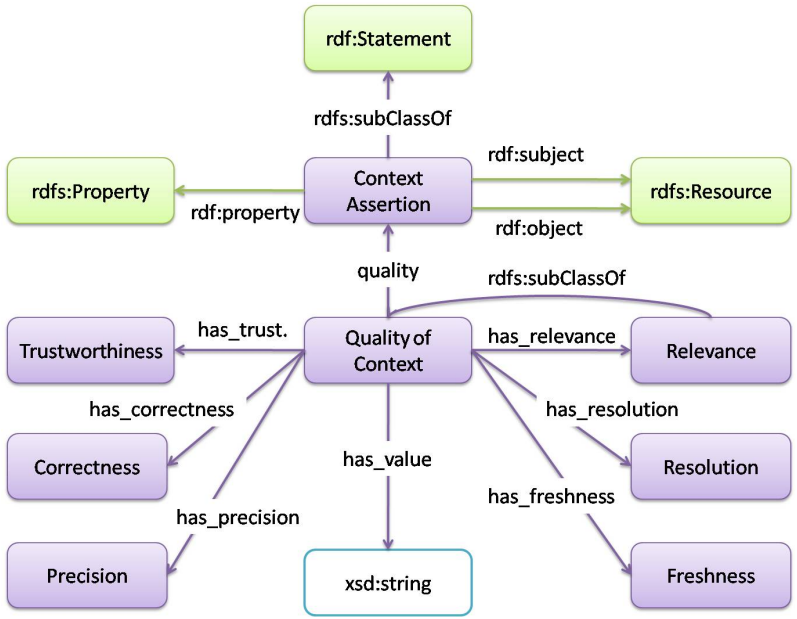
- Proteus associates a *quality attribute* to contexts and policies
- Each quality attribute includes a numerical value and several sub-attributes:
  - freshness, precision, correctness, trustworthiness, relevance, resolution

1. Context elements are associated to constraints on quality attributes (in addition to constraints on context attribute values)
   - Context assertions are associated to a certain value of quality attribute(s) (in addition to values for context attributes)

2. Policies are also associated to QoC constraints (thresholds)

QoC Representation Model

- QoC constraint representation → DL + extension to RDF reification
  - Integrated with Proteus v1.0 – uniform context/policy representation (DL)
  - Compatible with existing DL/RDF reasoners – compliant with RDF model



QoC-based Context Filtering

Example

Any context assertion about the property "located" must be provided with a quality value of 0.8 or above.

$$\text{QoC\_constraint} \equiv \text{CtxAssert} \sqcap \exists \text{ quality.QoC\_over0.8} \sqcap \exists \text{ rdf:property.}\{\text{located}\}$$

- Proteus filters out all context assertions (describing the current state) whose quality level does not satisfy quality constraints
- Context element filtering might depend on context sources and application domain – coarse-grained (from a security perspective)
  - Reducing the number of context assertions improves efficiency
  - Only "reliable enough" context assertions are considered to take security decisions

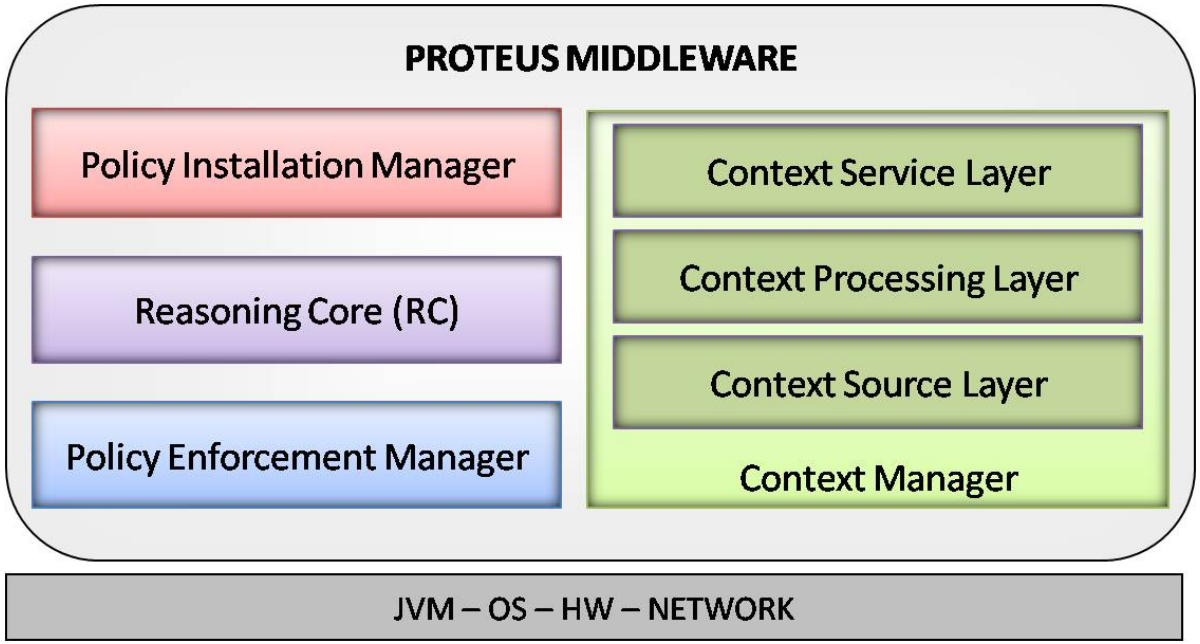
QoC-based Policy Evaluation

- Policies are assigned a quality threshold
- Current states are provided with different quality values (combining quality values of single context assertions)

- Proteus activates only policies matched by a current state that
  1. matches the policy protection context AND
  2. has a quality value higher than the policy threshold

- Policy thresholds might depend on sensitivity of the accessed resource, type of action, owner/requestor security properties
  - Variable levels of security can be obtained by adjusting quality thresholds

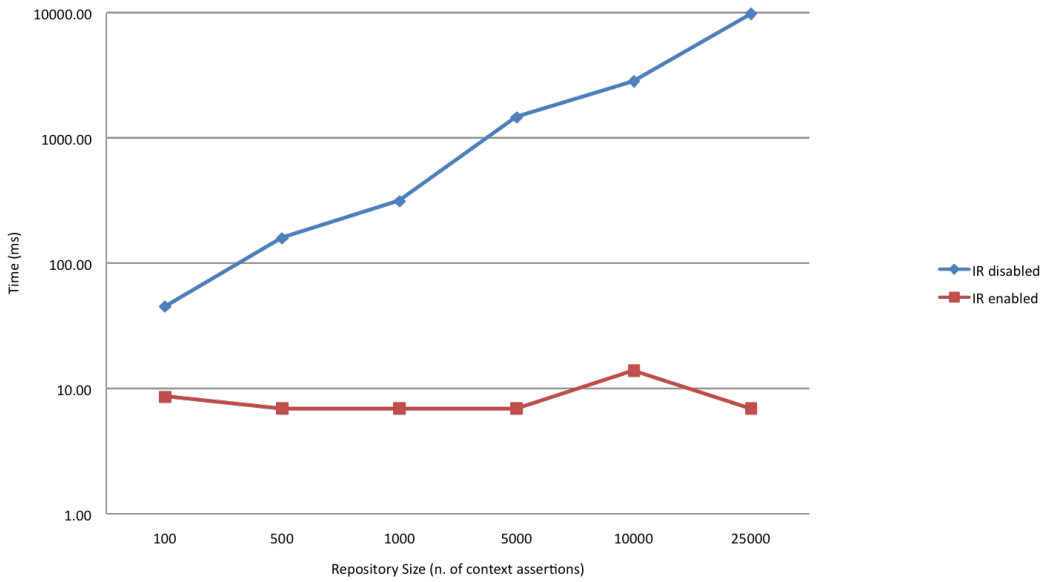
Proteus Middleware Architecture



- OWL-DL to represent contexts and policies
- Java first prototype extended to support QoC
  - Pellet 1.5 DL reasoner with incremental reasoning, via OWL API & SPARQL
  - Contory Context Source & Proteus Context Service modules

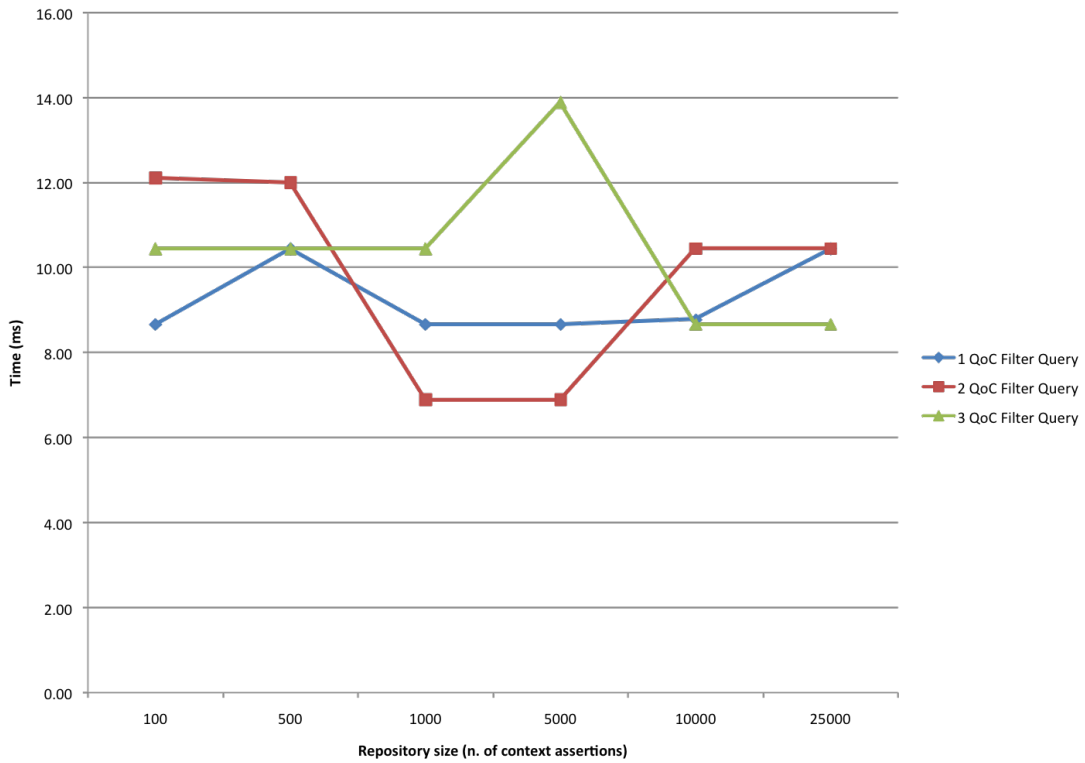
Prototype Evaluation – Context Repository Update

- Improved performance with Pellet incremental reasoning
  - Times to add context assertions show little dependence on repository size
    10 ms with 25k stored context assertions
  - No real advantage in context assertion removal



Prototype Evaluation – Context Query

- QoC constraints reduce query response time by limiting the number of retrieved context assertions
  - QoC-constrained queries not dependent on repository size
    $\longrightarrow$ increased efficiency



Outline

Summary

- Proteus is a QoC-aware access control framework
  - QoC modeling & reasoning explicitly account for QoC impact on security
  - QoC improves efficiency by discarding unreliable context information

- Ongoing & Future Work
  - QoC/context information management & storage optimizations
  - Integration with existing QoC calculation frameworks/tecnhiques
  - Evaluation of usability of the proposed approach