

Security Checks in Programming Languages for Ubiquitous Environments

PSPT2004, Workshop on Pervasive, Security, Privacy and Trust

Aug.26 2004

Authors : Eun-Sun Cho (CBNU), Kang-Woo Lee (ETRI)

Table of Contents

- Motivation ✓
- PLUE(a Programming Language for Ubiquitous Environment) ✓
- Access Control Policy Description ✓
- A Static Analysis for Rule Firing ✓
- Related Works
- Conclusions

Motivations

- Programs for ubiquitous computing often manage sensitive and/or private data.
- But, few of them deal with the security holes.
 - Security support seems to be tedious to the programmers.
 - And, it will mass up the programs!
- More effort needed for reduction on the security flaws in ubiquitous environments.

Introduction

- What we elaborated here
 - A security policy description method for access control to data in ubiquitous environment
 - Prediction on the rule evaluation - whether a specific rule can be fired when a given policy is applied

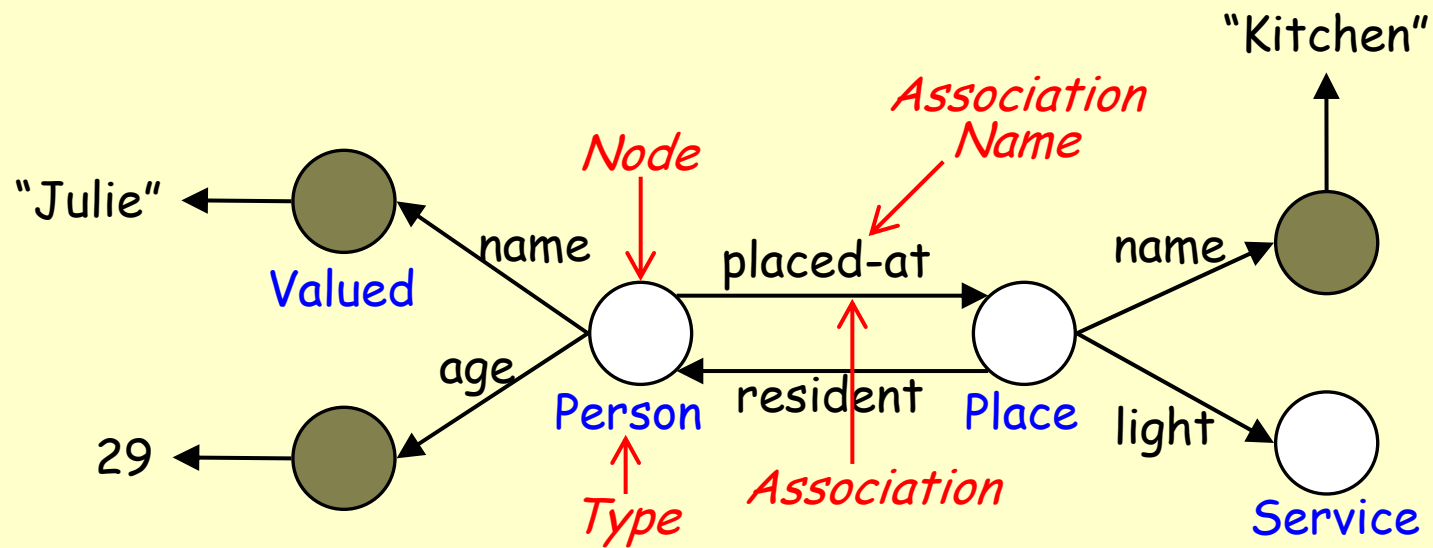
PLUE

(a Programming Language for
Ubiquitous Environment)

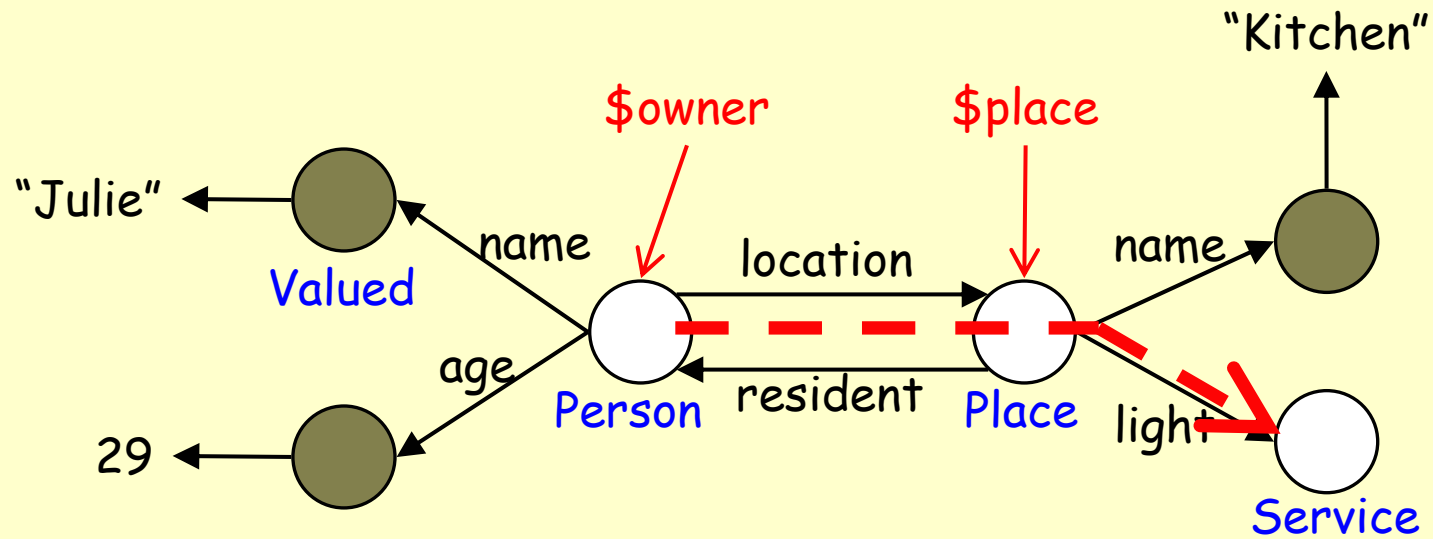
Overview of PLUE

- A Java extension for developing ubiquitous applications
 - A uniform data model for context
 - Event supports
 - (ECA) rule processing
 - CORBA service invocation
- Translated into Java classes with Jess rules

PLUE: Context Data Model



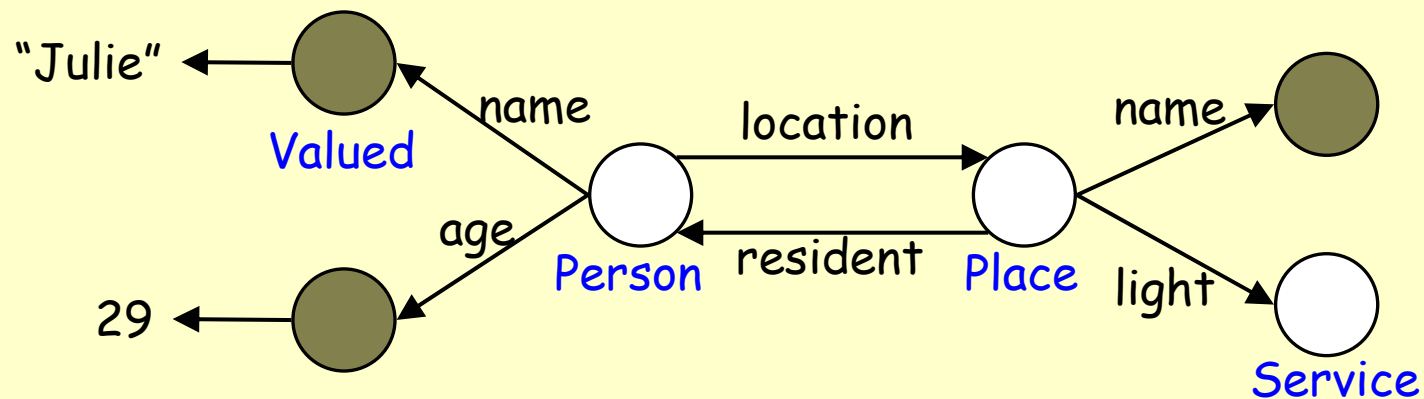
PLUE: Path Expressions



"\$owner.location.light"

PLUE: Events

- Some nodes raise events
 - Place-type: UserEntered, UserLeft
 - Valued-type: ValueChanged
 - Person-type: LocationChanged



PLUE: (ECA) Rule Processing

- Purpose
 - To be more smart (c.f. expert system)
 - To be reactive to environmental changes
- Exploit Jess rule engine internally

```
task SmartRoom {  
    // When a person entered a room,  
    // turn on the light of the room  
    on ($place.UserEntered e)  
    if ( $place.light.isOff() ) {  
        $place.light.turnOn();  
    }  
}
```

ECA Rule

Security Policies for PLUE

Access Control Policy Description : Basic Items

- Basic items

- (C, O, R)

- Who (C : *Credential*) will do Which action (R : *Right*) on What object (O : *Object*).

- Examples

- (Julie, HerROOM.light, B/M)

- (ANY, RESTROOM.light, M)

Access Control Policy Description :Quantified Items

- Quantified items
 - v_i in $template_i (C, O, R)$
 - ← For all $v_i \in template_i$, C will do R on O .
(v_i is a free variable for (C, O, R))
- Examples
 - h in Place ($ANY, h.light, B/M$)
 - P in Person ($p, p.preferred_temp.high, B/M$)
 - P in Person ($p, p.preferred_temp.low, B/M$)

A Static Analyzer for PLUE

A Static Analyzer for Rule Firing Prediction

- Any given security policy will mass up the program, and can make it difficult debugging it.

eg)

```
on ($place.UserEntered e) {  
  if ($place.temperature  
      > e.user.preferred_temp.high )  
    $place.air_conditioner.turnOn();  
}
```

- Can this rule be evaluated for everyone?
 - Can the air conditioner be turned on for anybody?
- How can we predict it?

A Static Analyzer for Rule Firing Prediction (cont's')

- An abstract interpreter gathers following information
 - *VarRights*: access rights and related objects necessary for access to each variable
 - *ExpRights*: access rights and related objects necessary for access to the expression that it currently interprets
 - *Allrights*: a list of access rights and related objects that required for a rule to be evaluated

A Static Analyzer for Rule Firing Prediction (conts')

- *The signature of the semantics function*

$$\begin{aligned} \varepsilon^\# : \text{Expr} &\rightarrow \text{Env}_\perp^\# \times \text{Heap}_\perp^\# \times \text{AllRights} \times \text{VarRights} \\ &\rightarrow (\text{ORef}^\# \times \text{ExpRights} \times \text{Env}_\perp^\# \times \text{Heap}_\perp^\# \times \text{AllRights} \times \\ &\quad \text{VarRights}) \end{aligned}$$

- Selected semantic rules

- $F^\# \varepsilon^\# [x] \langle \sigma^\#, h^\#, a^\#, z^\# \rangle =$

$$\langle \sigma^\#(x), z^\#(x), \sigma^\#, h^\#, a^\# \cup z^\#(x), z^\# \rangle$$

- $F^\# \varepsilon^\# [\text{path}.x] \langle \sigma^\#, h^\#, a^\#, z^\# \rangle =$

$$\text{let } \langle rs_1^\#, k_1^\#, \sigma_1^\#, h^\#, a_1^\#, z_1^\# \rangle =$$
$$\varepsilon^\# [\text{path}] \langle \sigma^\#, h^\#, a^\#, z^\# \rangle$$
$$\text{in } \langle \{h^\#(r_1^\#) \mid r_1^\# \in rs_1^\#\}, k_1^\# \cup \{\langle \text{path}.x, B \rangle\}, \\ \sigma^\#, h^\#, a_1^\# \cup k_1^\# \cup \{\langle \text{path}.x, B \rangle\}, z^\# \rangle$$

A Static Analyzer for Rule Firing Prediction (cont's')

- **Theorem 2** A given credential c and a given policy p , the rule R is expected to be fired, which is denoted by $ExF(R, c, p)$ if and only if,

$$S_{ar}[a] \subseteq \bigcup_{\langle c, o, t \rangle \in p} S_p[\langle o, t \rangle],$$

where

- a : an AllRights, derived from the abstract interpretation on R
- S_{ar} : the semantics function for AllRights
- S_p : the semantics function for the policy

"The static analyzer expects that R requires only those permitted to c under p "

A Static Analyzer for Rule Firing Prediction (conts')

- **Theorem 3** For a given credential c , a set $\{R/ExF(R,c,p)\}$ extracts all the rules expected to be fired according to the policy p .
- **Theorem 4** For a given rule R , a set $\{c/ExF(R,c,p)\}$ extracts all the credentials that is expected to fire the rule R .

Related Works

- Various efforts on security support have been made for ubiquitous environments
 - But rare of them considered programming facilities, and security flaw reduction
- Works on database policy description does not cover so far
 - Service invocation
 - Quantification and privacy
- Works on information flow detection have never been applied to ubiquitous data and programs yet.

Conclusions and Future Works

- Toward better programming environment for reduction on the security flaws in ubiquitous environments
 - A method of security policy description for access control to data in ubiquitous environment
 - A static checker to predict on the rule evaluation - if the rule can be fired when a given policy is applied
- Future Works
 - Finding out effective algorithms for the theorems
 - Refining the semantics and the policy model

Thank You.

(eschough@cbnu.ac.kr)