Dependency Path Patterns as the Foundation of Access Control in Provenance-aware Systems

June 14, 2012
TaPP’12

Dang Nguyen, Jaehong Park and Ravi Sandhu
Institute for Cyber Security
University of Texas at San Antonio
Access control in Provenance-aware Systems

• Provenance Access Control (PAC)
  – Controlling access to provenance data which could be more sensitive than the underlying data
  – Needs access control models/mechanisms (e.g., RBAC)
  – (Meaningful) control granularity?

• Provenance-based Access Control (PBAC)
  – Using provenance data to control access to the underlying data
  – Provenance-based policy specification

Meaningful granularity of provenance data?
PAC & PBAC in Applications

• Common Foundation
  – Base provenance data
  – Dependency list
    • Dependency Name: meaningful, named abstraction
    • matching regular expression-based causality dependency path pattern

• PAC and PBAC are complementary
  – In PAC, control decision can be based on provenance data (PB-PAC)
  – In PBAC, PAC can be used for added trustworthiness on provenance data
Provenance Data

• Directed Acyclic Graph (DAG)

• Causality dependencies between entities (acting users, action processes and data objects)

• Dependency graph can be traced for extracting pedigree, usage, versioning information, etc.
  • PBAC can support origin/usage-based control, Dynamic Separation of Duty (DSOD), workflow control, etc.
From Open Provenance Model (OPM)

- **3 Nodes**
  - Artifact (ellipse)
  - Process (Rectangle)
  - Agent (Hexagon)

- **5 Causality dependency edges** (not dataflow)

- **Provenance data**: a set of 2 entities & 1 dependency
  - E.g., \((ag, p1, a1, a2)\): \(<p1, ag, c>, <p1, a1, u>, <a2, p1, g>\)
Direct vs. Indirect Dependencies

• **Direct dependencies**
  – Used (u), wasGeneratedBy (g), wasControlledBy (c)
  – Captured from transactions as base provenance data

• **Indirect dependencies**
  – System-computable dependencies
    • using pre-defined dependency names and matching dependency path patterns
  – User-declared dependencies
    • using pre-defined dependency names
Object Dependency List (DL\textsubscript{O})

• A set of pairs of
  – abstracted dependency names (DNAME) and
  – regular expression-based object dependency path patterns (DPATH)

• Examples
  – \(< \text{wasSubmittedVof}, g_{\text{submit}\cdot u_{\text{input}}} >
  – \(< \text{wasAuthoredBy},
    \text{wasSubmittedVof}\cdot \text{wasReplacedVof} \cdot g_{\text{upload}\cdot c} >

PBAC vs. PAC

PBAC
- DNAME based Policy
- User Authorization
- Action Validation
- System-computable dependency
- User-declared dependency
- Policy Retrieval

PAC
- DNAME as an initial control unit
- Sanitization/filtering on initial trace results
- Needs additional access control system

Base Provenance Data & Dependency List

Provenance-aware System
PBAC Models
Example: A Homework Grading System

1. Anyone can upload a homework.
2. A user can replace a homework if she uploaded it (origin-based control) and the homework is not submitted yet.
3. A user can submit a homework if she uploaded it and the homework is not submitted already. (workflow control)
4. A user can review a homework if she is not the author of the homework (DSOD), the user did not review the homework earlier, and the homework is submitted already but not graded yet.
5. A user can grade a homework if the homework is reviewed but not graded yet.
Sample Transactions &
Base Provenance Data

- \((au1, upload1, o_{1v1})\): \(<\text{upload1, au1, c}>,<o_{1v1,upload1},g_{upload}>\)
- \((au1, replace1, o_{1v1}, o_{1v2})\): \(<\text{replace1, au1, c}>,<\text{replace1, o}_{1v1},\text{u_input}>,<o_{1v2,\text{replace1}},g_{\text{replace}}>\)
- \((au1, submit1, o_{1v2}, o_{1v3})\): \(<\text{submit1, au1, c}>,<\text{submit1,o}_{1v2},\text{u_input}>,<o_{1v3,\text{submit1}},g_{\text{submit}}>\)
- \((au2, review1, o_{1v3}, o_{2v1})\): \(<\text{review1, au2, c}>,<\text{review1,o}_{1v3},\text{u_input}>,<o_{2v1,\text{review1}},g_{\text{review}}>\)
- \((au3, grade1, o_{1v3}, o_{3v1})\): \(<\text{grade1, au3, c}>,<\text{grade1,o}_{1v3},\text{u_input}>,<o_{3v1,\text{grade1}},g_{\text{grade}}>\)
A Sample Base Provenance Data
A Sample Base Provenance Data

wasReviewedOby

wasReviewedOof

wasGradedOof

wasSubmittedVof

DL₀: < wasReplacedVof, g_replace・u_input >
A Sample Base Provenance Data

\[ DL_0: \langle \text{wasAuthoredBy}, \text{wasSubmittedVof}\rangle, \text{wasReplacedVof} \ast \ast_g_{\text{upload}.c} \]
A Sample Base Provenance Data

\[ DL_0: \langle \text{wasReviewedBy}, \text{wasReviewedOof}^{-1}. \text{wasReviewedOby} \rangle \]
Sample Object Dependency List (DL₀)

1. < wasReplacedVof, g_replace.u_input >
2. < wasSubmittedVof, g_submit.u_input >
3. < wasReviewedOof, g_review.u_input >
4. < wasReviewedOby, g_review.c >
5. < wasGradedOof, g_grade.u_input >
6. < wasAuthoredBy, wasSubmittedVof?.wasReplacedVof *.g_upload.c >
7. < wasReviewedBy, wasReviewedOof⁻¹. wasReviewedOby >
Sample Policies

1. Anyone can upload a homework.
2. A user can replace a homework if she uploaded it (origin-based control) and the homework is not submitted yet.
3. A user can submit a homework if she uploaded it and the homework is not submitted already. (workflow control)

1. allow(au, upload, o) ⇒ true
2. allow(au, replace, o) ⇒ auє(o, wasAuthoredBy)
∧ |(o,wasSubmittedVof)| = 0.
3. allow(au, submit, o) ⇒ auє(o, wasAuthoredBy)
∧ |(o,wasSubmittedVof)| = 0.
Sample Policies (cont.)

4. A user can review a homework if she is not the author of the homework (DSOD), the user did not review the homework earlier, and the homework is submitted already but not graded yet.

5. A user can grade a homework if the homework is reviewed but not graded yet.

4. \( \text{allow}(au, \text{review}, o) \Rightarrow au \not\in (o, \text{wasAuthoredBy}) \land au \not\in (o, \text{wasReviewedBy}) \land |(o, \text{wasSubmittedV of})| \neq 0 \land |(o,\text{wasGradedOof}^{-1})| = 0. \)

5. \( \text{allow}(au, \text{grade}, o) \Rightarrow |(o, \text{wasReviewedOof})| \neq 0 \land |(o,\text{wasGradedOof}^{-1})| = 0. \)
Summary

• Regular expression-based dependency path pattern

• Introduced the notion of named abstractions of causality dependency path patterns as a foundation for PBAC and PAC

• Supports Simple and effective policy specification and access control management

• Supports DSOD, workflow control, origin-based control, usage-based control, object versioning, etc.
What’s next?

- Enhancing/extending PBAC model
- Provenance Access Control Models
- Provenance data sharing in multiple systems
Thank you

• Questions and Comments?