Provenance-based Access Control Models

July 31, 2014

Dissertation Defense

Dang Nguyen

Institute for Cyber Security

University of Texas at San Antonio

Presentation Outline

1. Introduction
2. Provenance Data Model
3. Provenance-based Access Control Models
4. PBAC Architecture in Cloud Infrastructure-as-a-Service
5. Conclusion

Background: what is provenance?

Art definition of provenance

- Essential in judging authenticity and evaluating worth.

Data provenance in computing systems

- Is different from log data.
- Contains linkage of information pieces.
- Is utilized in different computing areas.

Access Control Challenges

- Usability of provenance
  - *Capturing,*
  - *Storing,*
  - and *Querying* provenance data.
- Utility of provenance
  - *Policy specification,*
  - *Evaluation,*
  - and *Enforcement*.
- Provenance in cloud environment
  - Tenant-awareness

Provenance Data Model

Provenance-based Access Control

PBAC in IaaS Architecture

Security of provenance: provenance access control

Access Control Approaches

- Traditional access control
  - Based on single units of control: roles, primitive attributes, etc.
- Relationship-based access control
  - Graph-based.
  - Does not make use of history information.
- Based on history information
  - Utilizes log data to extract useful information
    - Mainly looks at users' history.
  - Cannot specify access control based on linkage information.
  - Assume history information is readily available.

Provenance-based Access Control

Provenance-based Access Control (PBAC)

- So far, no **comprehensive** and **well-defined** model in the literature.
- Compared to other access control approaches, **PBAC** provides **richer** access control mechanisms
  - Finer-grained policy and control.
  - Provides effective means of history information usage.
- **Easily configured** to apply in different computing domains and platforms
  - Single system (XACML)
  - Multi-tenant cloud (OpenStack)

Contributions

- Proposed a provenance data model which enables PBAC configurations in multiple application domains.
- Proposed provenance-based access control models which provides enhanced and finer-grained access control features.
  - Implemented and evaluated an XACML-extended prototype.
- Proposed architecture to enable PBAC in cloud IaaS.
  - Implemented and evaluated an OpenStack-extended prototype.

Thesis Statement

Provenance data forms a directed-acyclic graph where graph edges exhibit the causality dependency relations between graph nodes that represent provenance entities.

A provenance data model that can enable and facilitate the capture, storage and utilization of such information through regular expression based path patterns can provide a foundation for enhancing access control mechanisms.

In essence, provenance-based access control models can provide effective and expressive capabilities in addressing access control issues, including traditional and previously not discussed dynamic separation of duties, in single systems, distributed systems, and within a single tenant and across multiple tenants cloud environment.

Scope and Assumptions

- Assumptions
  - Provenance data is uncompromised and protected.
  - Provenance data is correct.
  - *Provenance of provenance is not considered.*

- Experimental Scope
  - Does not include provenance capture.
  - Does not include concurrent, dependent access requests.

Presentation Outline

1. Introduction
2. Provenance Data Model
3. Provenance-based Access Control Models
4. PBAC Architecture in Cloud Infrastructure-as-a-Service
5. Conclusion

Characteristics of Provenance Data

- Information of operations/transactions performed against data objects and versions
  - Actions that were performed against data
  - Acting Users/Subjects who performed actions on data
  - Data Objects used for actions
  - Data Objects generated from actions
  - Additional Contextual Information of the above entities

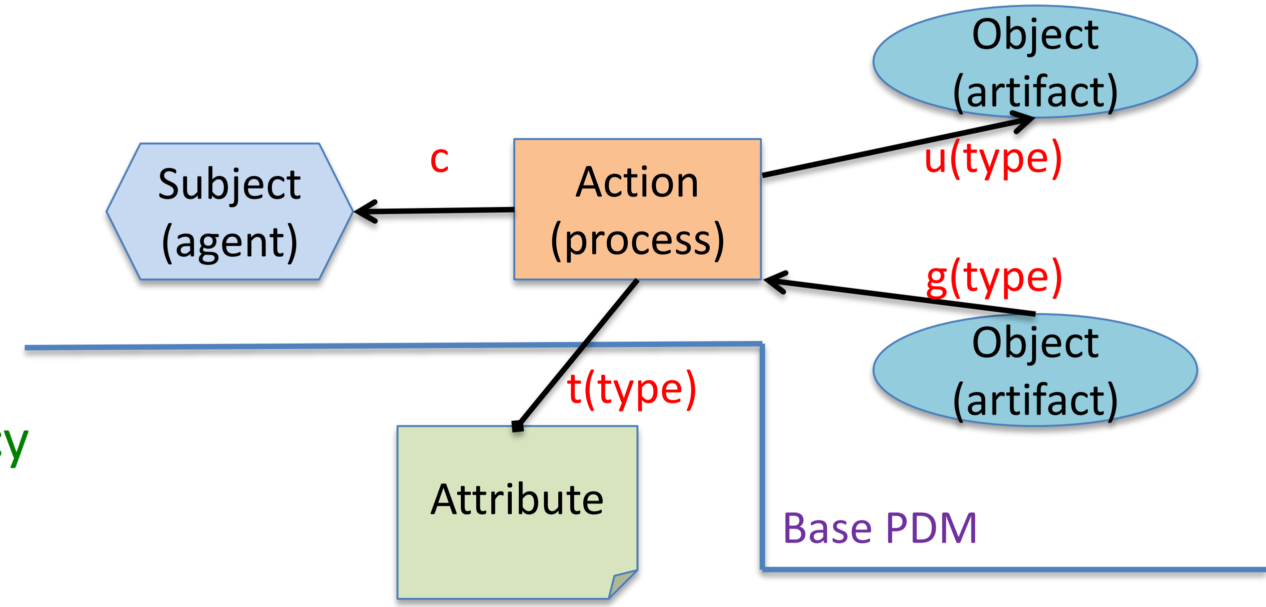
- Directed Acyclic Graph (DAG)
- Causality dependencies between entities (acting users / subjects, action processes and data objects)

- Dependency graph can be traced/traversed for the discovery of Origin, usage, versioning info, etc.

Provenance Data Model

[inspired by OPM]

- 4 Node Types
  - Object (Artifact)
  - Action (Process)
  - Subject (Agent)
  - Attribute
- 3 Causality dependency edge Types
  (not a dataflow) and
  Attribute Edge



Contextual Extension

c	wasControlledBy
u	used
g	wasGeneratedBy
t	hasAttribute

→ Dep. edge

→ Attrb. edge

Inverse edges are enabled for usage in queries, but **cycle-avoidant**.

Capturing, Storing, and Querying Provenance Data

Transaction:

(Subject1, Grade1, HW1, GradedHW1, ContextualInfoSet-Grade1)

capturing

RDF Triples:

(Grade1, u, HW1)
(Grade1, c, Subject1)
(GradedHW1, g, Grade1)

(Grade1, t[actingUser], Alice)
(Grade1, t[activeRole], TA)
(Grade1, t[weight], 2)
(Grade1, t[object-size], 10MB)

storing

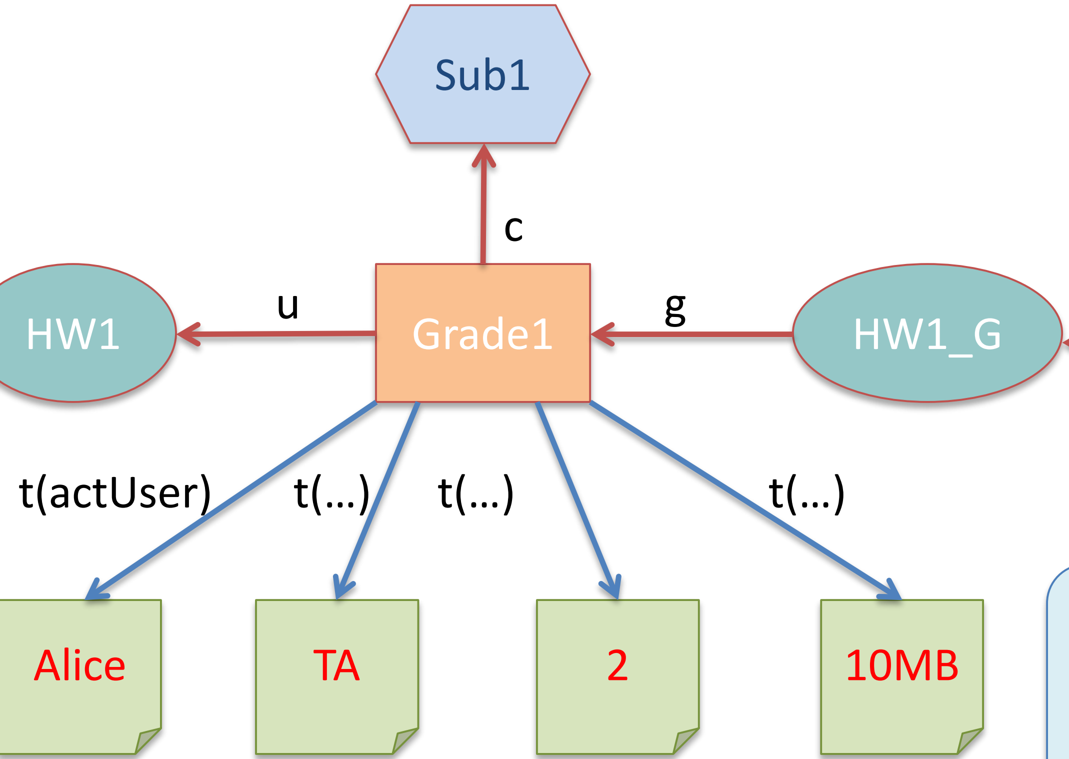
querying

querying

SPARQL:

SELECT ?agent WHERE { HW1_G [g:c] ?agent}
SELECT ?user WHERE { HW1_G [g:t[actUser]] ?user}

Provenance Graph Example



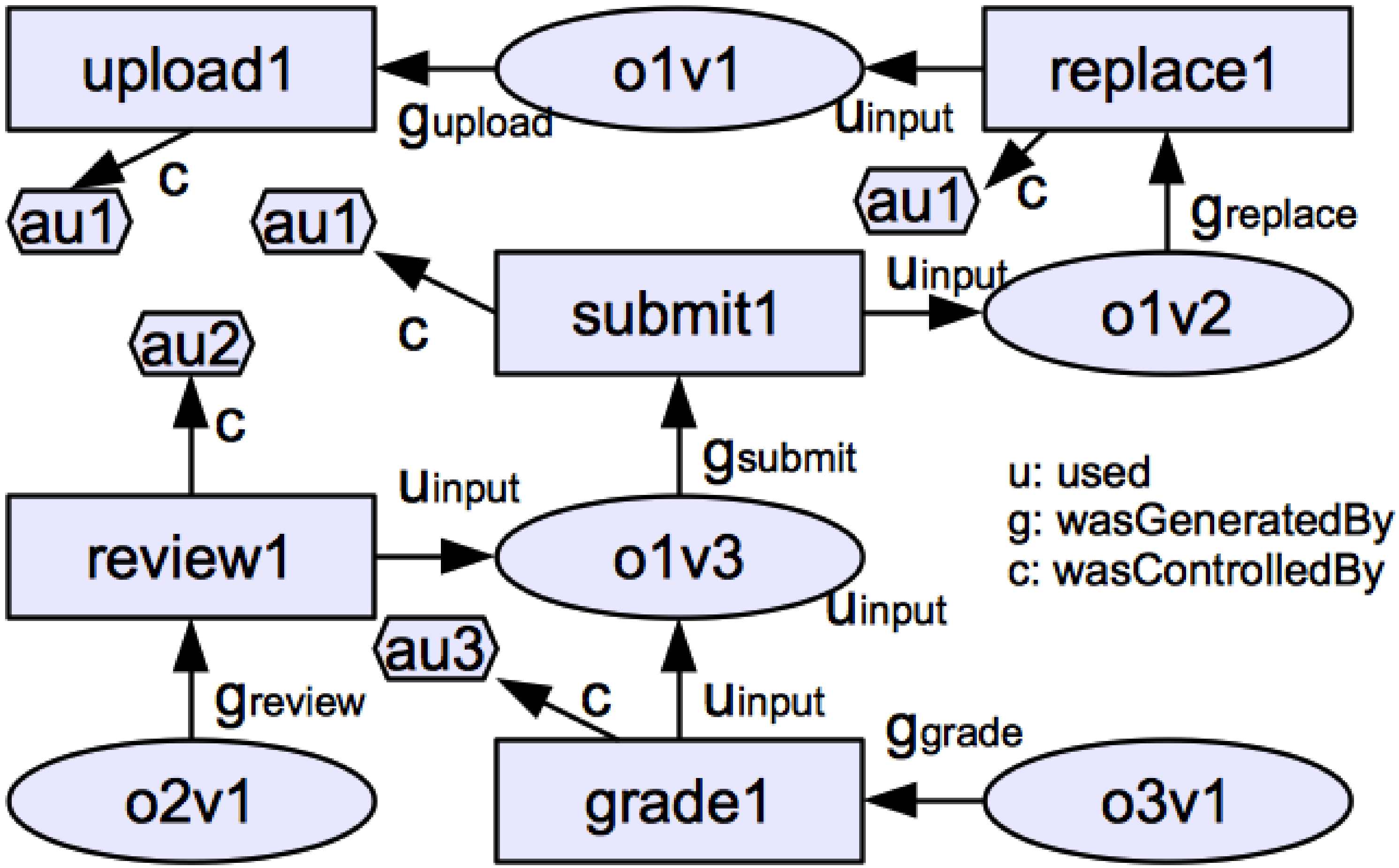
SELECT ?user
WHERE
{ HW1_G’ [g:u:g:c] ?user}

{ HW1_G’ *[[g:u]*:g:c]* ?user}

Study Case:
Homework Grading System

Students can **upload** a homework to the system, after which they can **replace** it multiple times before they **submit** the homework. Once it is **submitted**, the homework can be **reviewed** by other students or designated graders until it is **graded** by the teaching assistant (TA).

A Base Provenance Data Graph

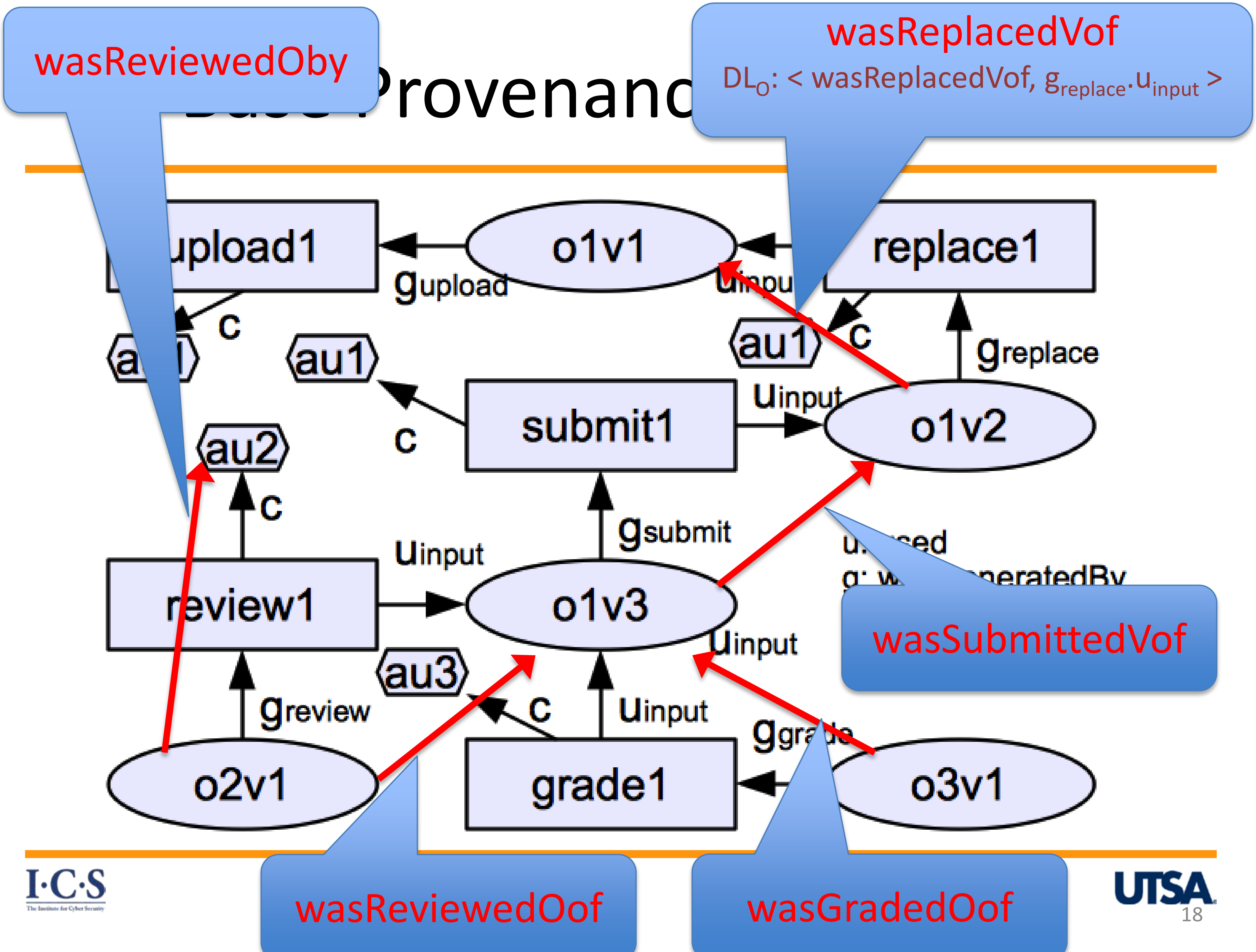


Dependency List

- Dependency List (DL): A set of identified dependencies that consists of pairs of
  - Dependency Name: abstracted dependency names (DNAME) and
  - regular expression-based dependency path pattern (DPATH)

- Examples
  - < wasReplacedVof, $g_{replace}.u_{input}$ >
  - < wasAuthoredBy, wasSubmittedVof?.wasReplacedVof $*.g_{upload}.c$ >

Provenanc



wasReviewedOby
wasReplacedVof
DL_O: < wasReplacedVof, $g_{replace}.u_{input}$ >
upload1
o1v1
replace1
g_{upload}
u_{input}
au1
c
au1
c
$g_{replace}$
au1
submit1
u_{input}
o1v2
c
au2
c
g_{submit}
review1
u_{input}
o1v3
wasSubmittedVof
u_{input}
au3
g_{review}
c
u_{input}
g_{grade}
o2v1
grade1
o3v1
wasReviewedOof
wasGradedOof

Presentation Outline

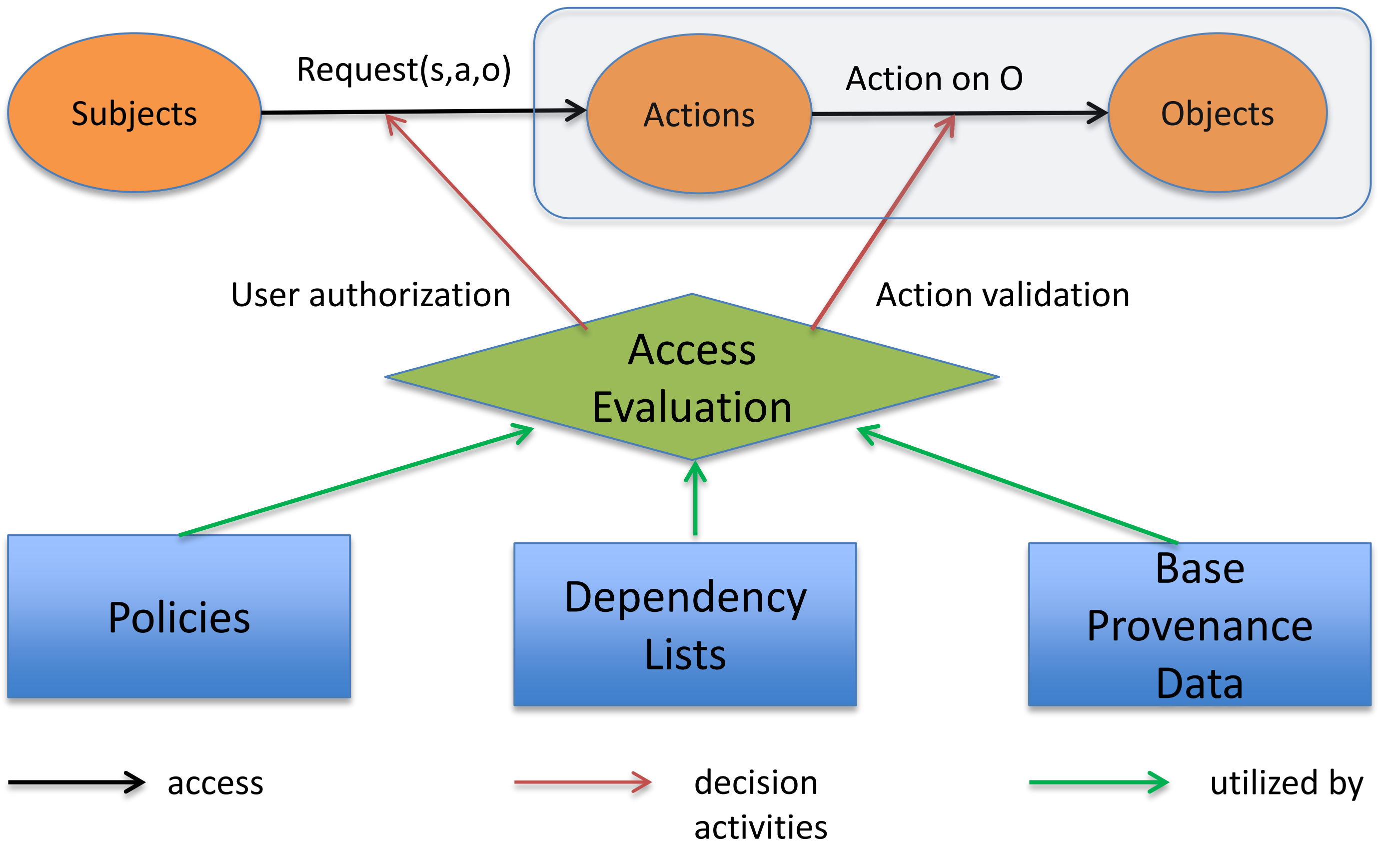
1. Introduction
2. Provenance Data Model
3. Provenance-based Access Control Models
4. PBAC Architecture in Cloud Infrastructure-as-a-Service
5. Conclusion

PBAC Models

- **$PBAC_B$**: utilizes base data model
  - Does not capture contextual information

- **$PBAC_C$**: extending the base model
  - Incorporate *contextual information* associated with the main entities (**Subjects**, etc.)
  - Extend base data model with attributes

PBAC$_B$ Components

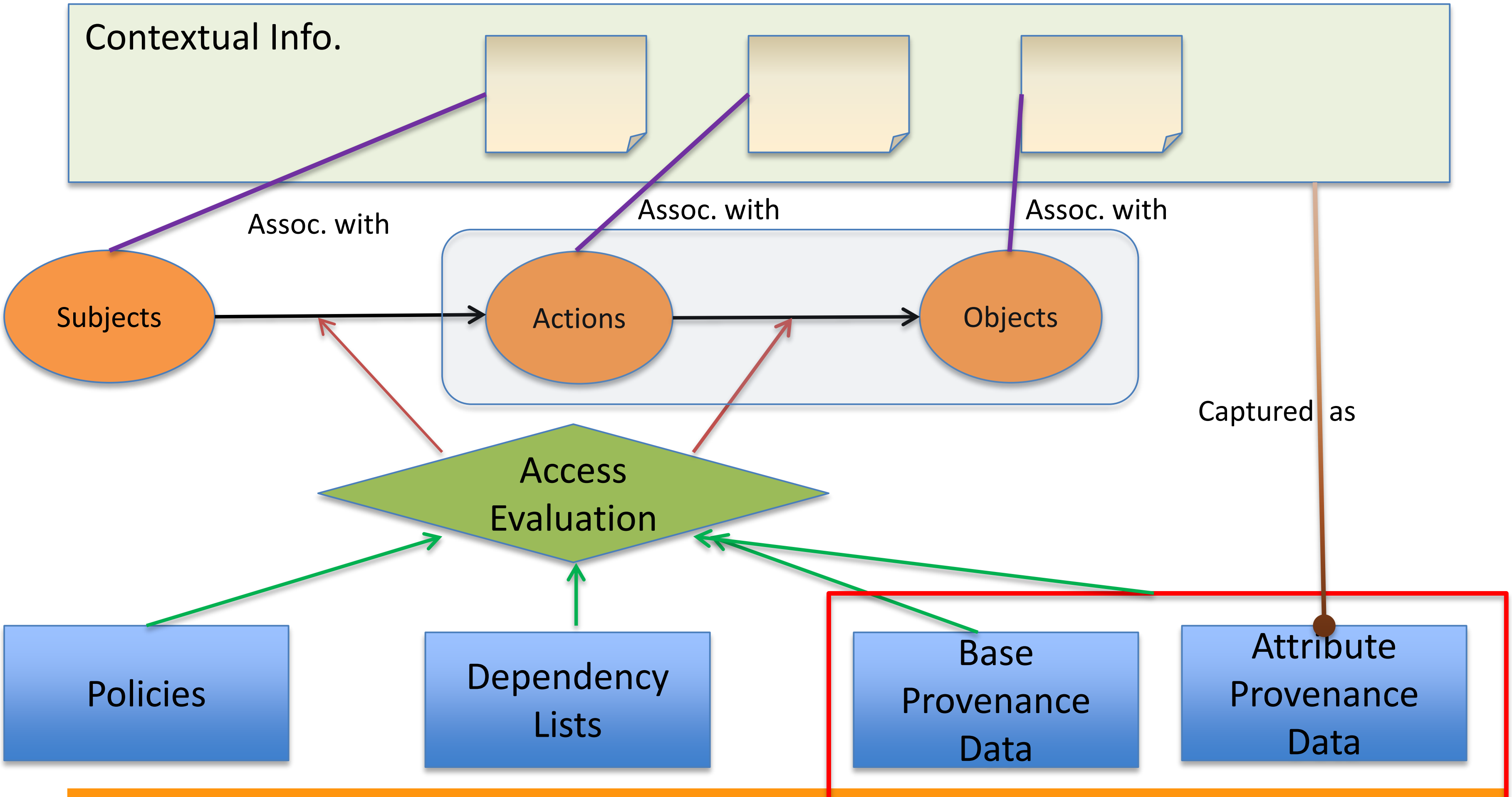


Sample Policies

1. Anyone can upload a homework.
2. A user can replace a homework if she uploaded it (usr. authz) and the homework is not submitted yet (act. valid) .

1. allow(au, upload, o) ⇒ true
2. allow(au, replace, o) ⇒ au∈(o, wasAuthoredBy) ∧|(o,wasSubmittedVof)| = 0.

$PBAC_C$ Components



DSOD Examples in HGS

- **Sample English policies**:
  - A student cannot **review** the homework he **submitted** – Object-based DSOD
  - A student cannot **grade** a homework before it is **submitted** – History-based DSOD
  - A student cannot **grade** a homework unless reviews' combined weights exceeds 3 – Transaction Control Expression

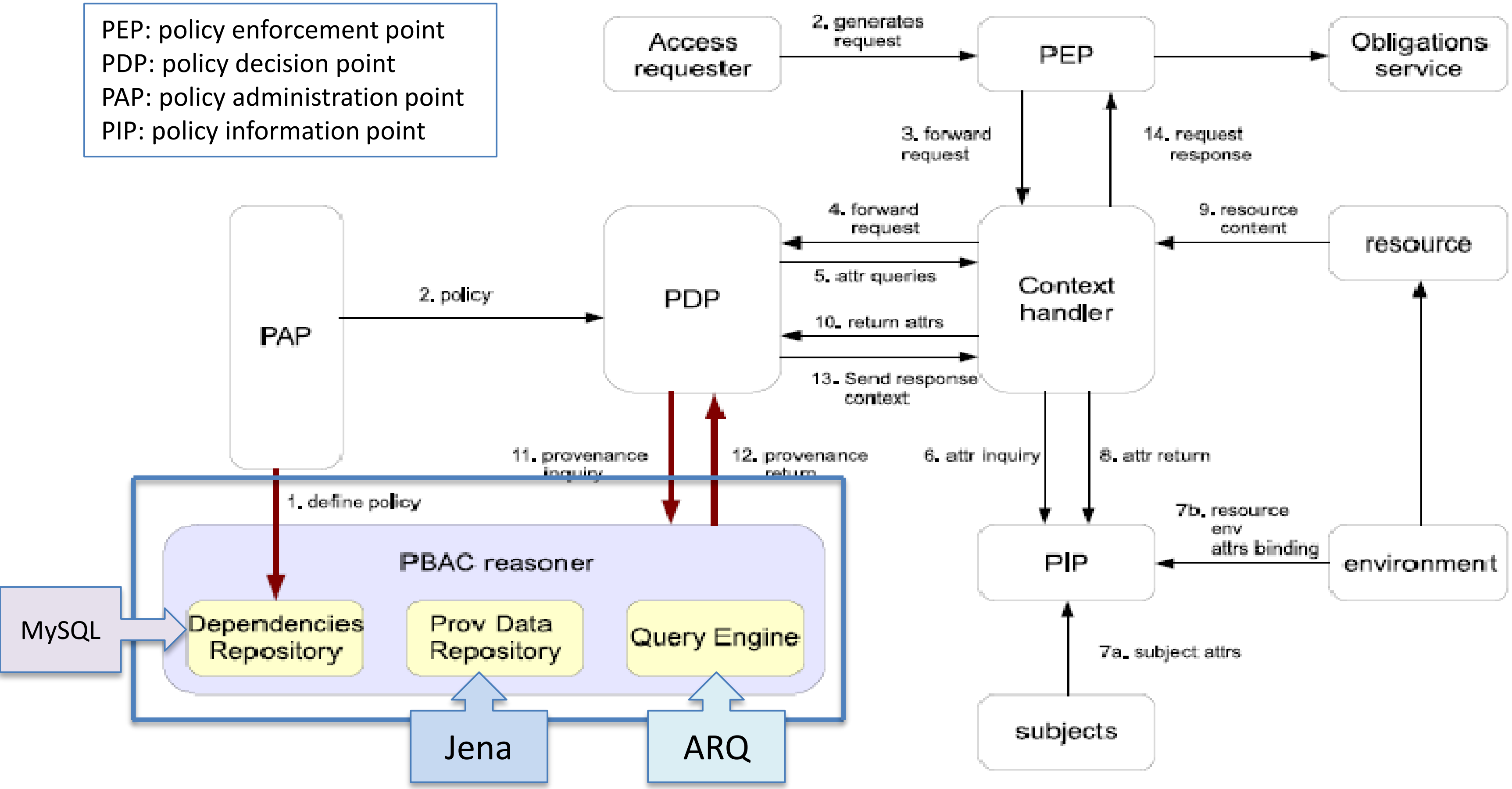
- **An informal policy**:

  allow(sub,grade,o) =>
  sum(o,previousReviewProcesses.hasAttributeOf(Weight)) <= 3

- Compatible to *XACML* policy language
  - Extending *OASIS XACML* architecture and implementation.

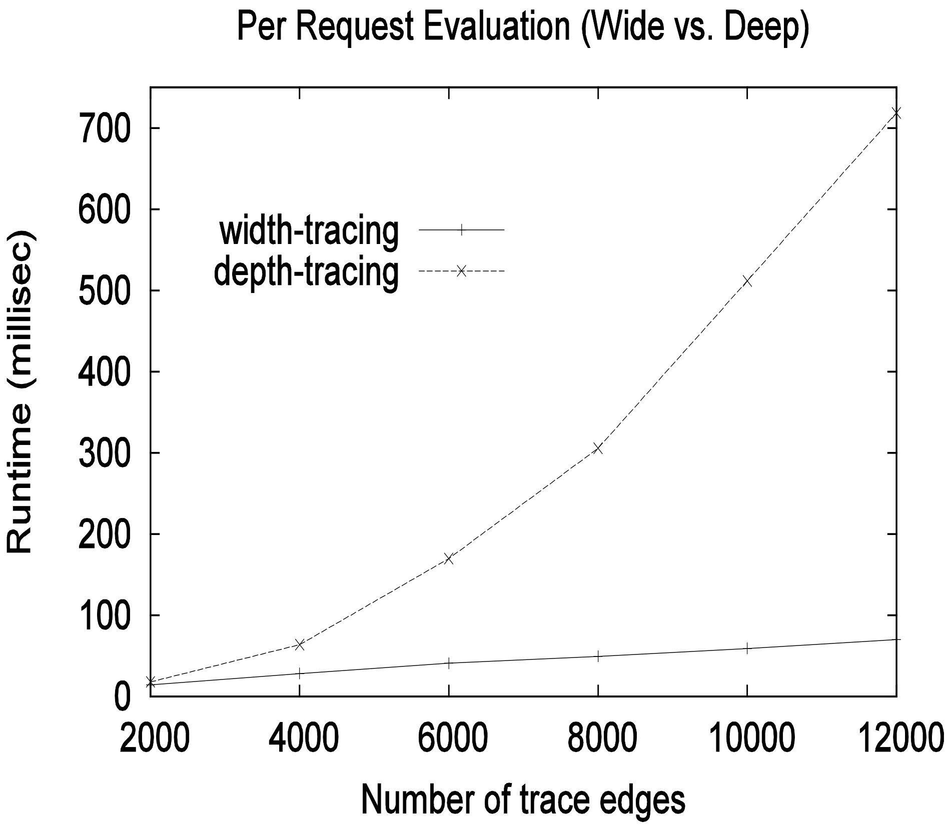
Extended XACML Architecture

PEP: policy enforcement point
PDP: policy decision point
PAP: policy administration point
PIP: policy information point



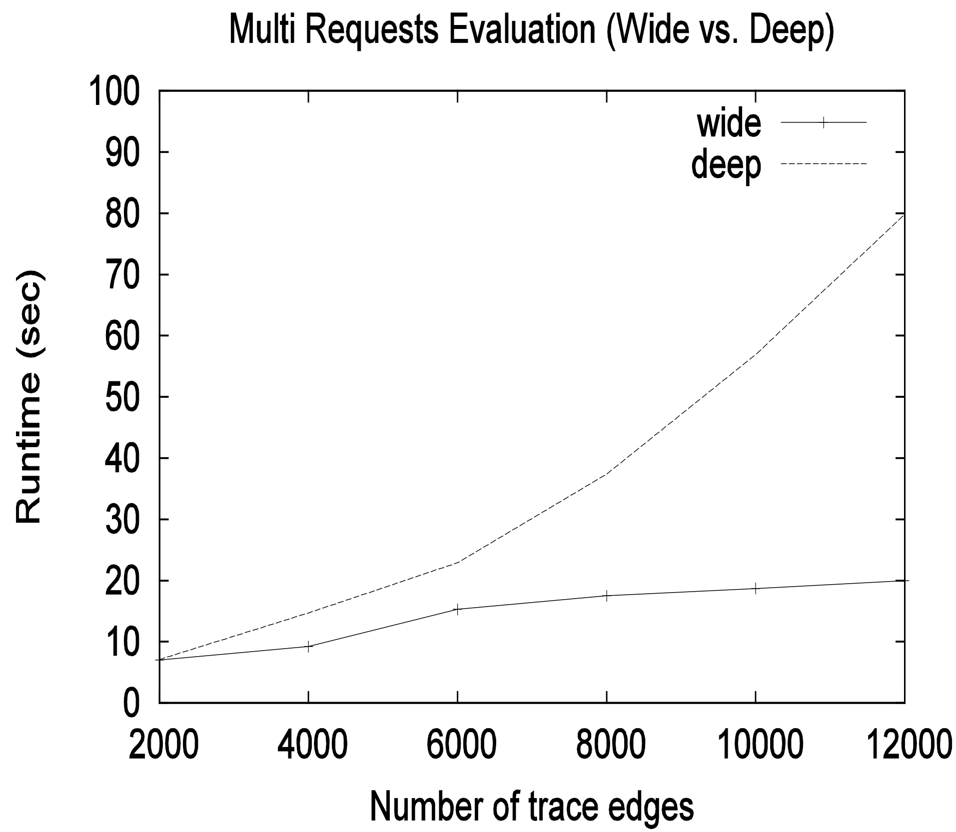
Experiment and Performance

- System
  - Ubuntu 12.10 image with 4GB Memory and 2.5 GHz quad-core CPU running on a Joyent SmartData center (ICS Private Cloud).
- Mock Data simulating HGS scenario
  - Extreme depth and width settings for graph traversal queries.
- Results for tracing 2k/12k edges
  - 0.017/0.718 second per deep request
  - 0.014/0.069 second per wide request



Throughput Evaluation

- 500 concurrent, nondependent requests
- Results for tracing 2k/12k edges
  - 0.014/0.16 second per deep request
  - 0.014/0.04 second per wide request



Presentation Outline

1. Introduction
2. Provenance Data Model and Access Control
3. Provenance-based Access Control Models
4. PBAC Architecture in Cloud Infrastructure-as-a-Service
5. Conclusion

Cloud Computing

- Cloud computing has been the “next big thing.”
- Has 3 primary service models:
  – Software-as-a-Service (SaaS)
  – Platform-as-a-Service (PaaS)
  – Infrastructure-as-a-Service (IaaS)
- We focus on PBAC for IaaS
  – Specifically, multi-tenant single-cloud systems.

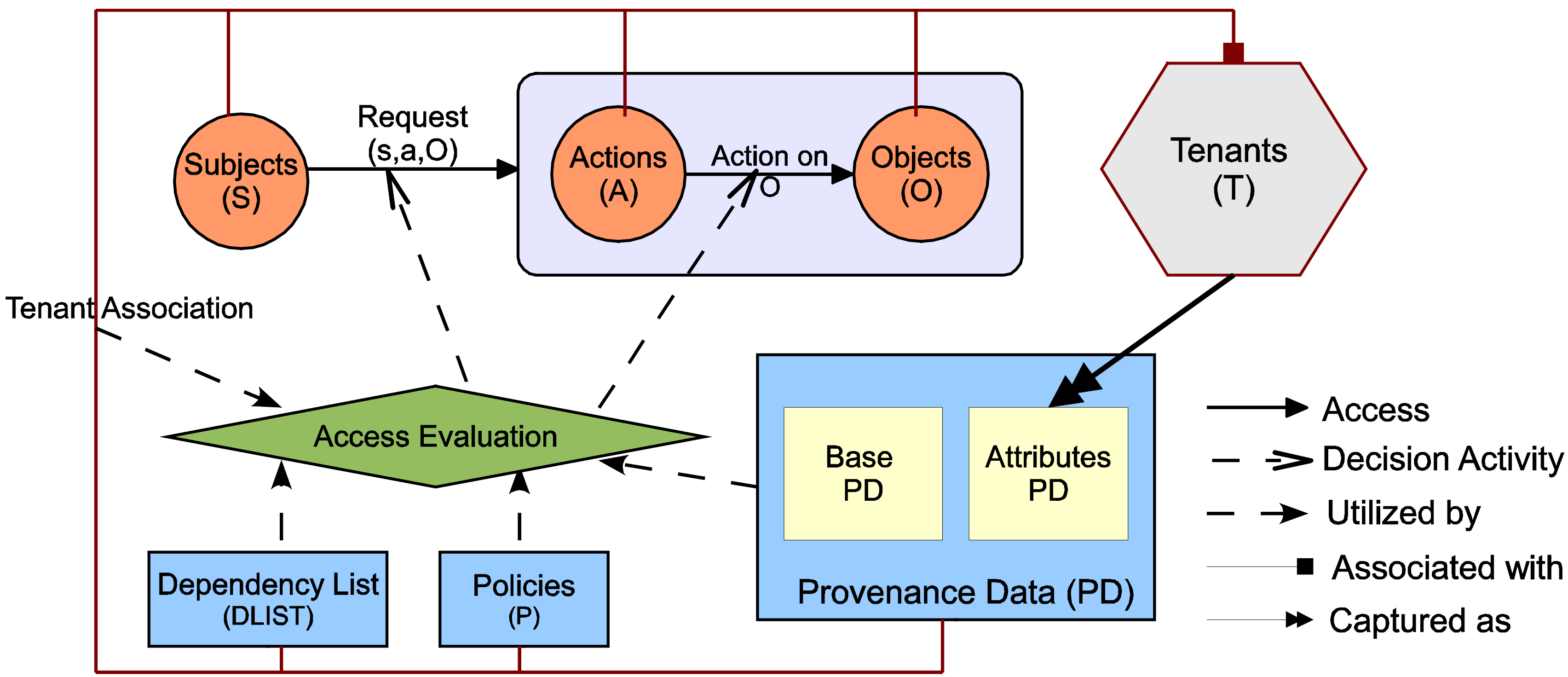
Access Control Aspects

- DSOD concerns for virtual resources management and protection
  - Ex: Only virtual images up-loaders are allowed to delete.

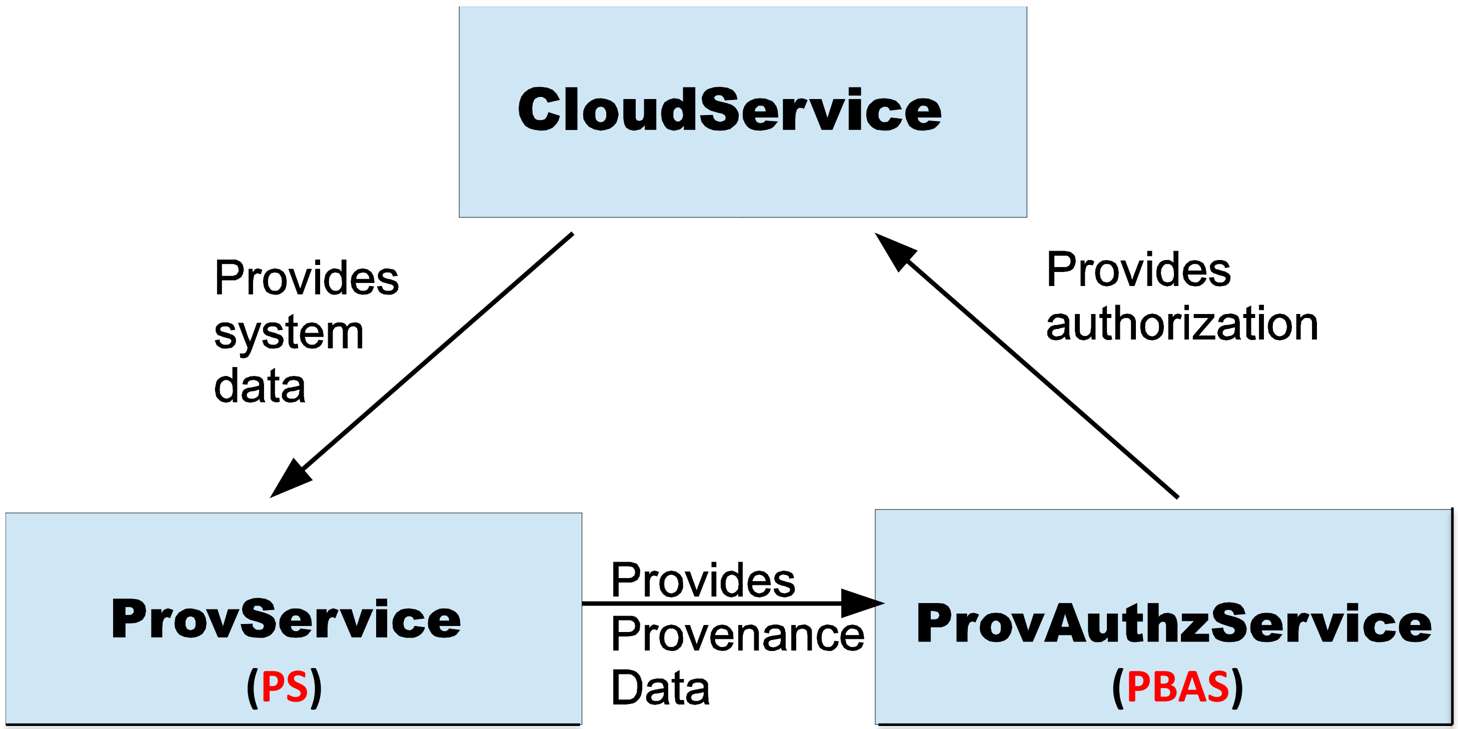
- Multi-tenant concerns
  - A virtual image may be created in one tenant, copied to another tenant and modified, and used to launch a virtual machine instance in another.

Tenant-aware PBAC

Tenants as contextual information.



Architecture Overview



Deployment Architecture

Variations:

- Integrated Deployment
- Stand-alone Deployment
- Hybrid Deployment

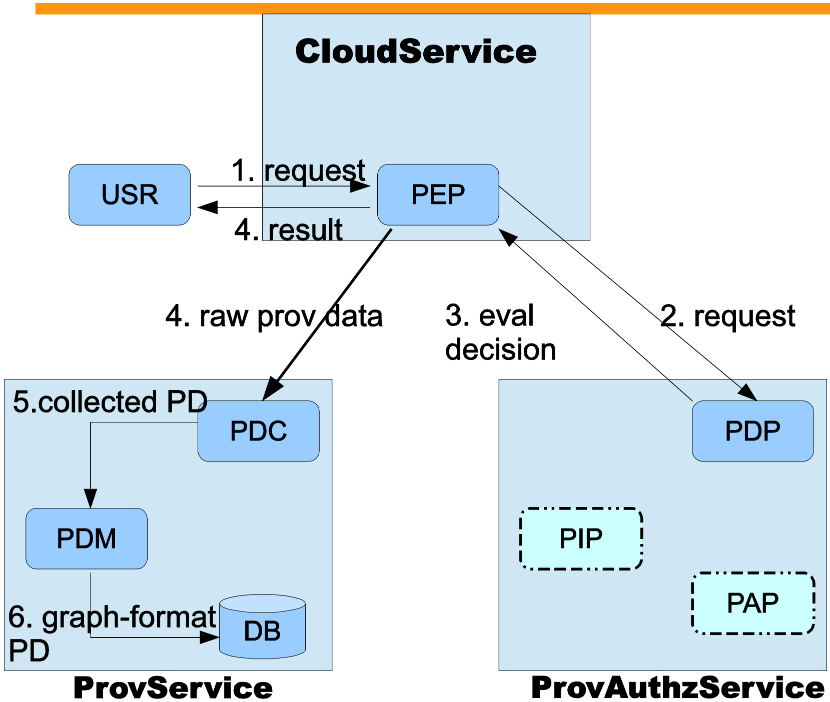
Design pros & cons:

Ease of integration -

Communication latency -

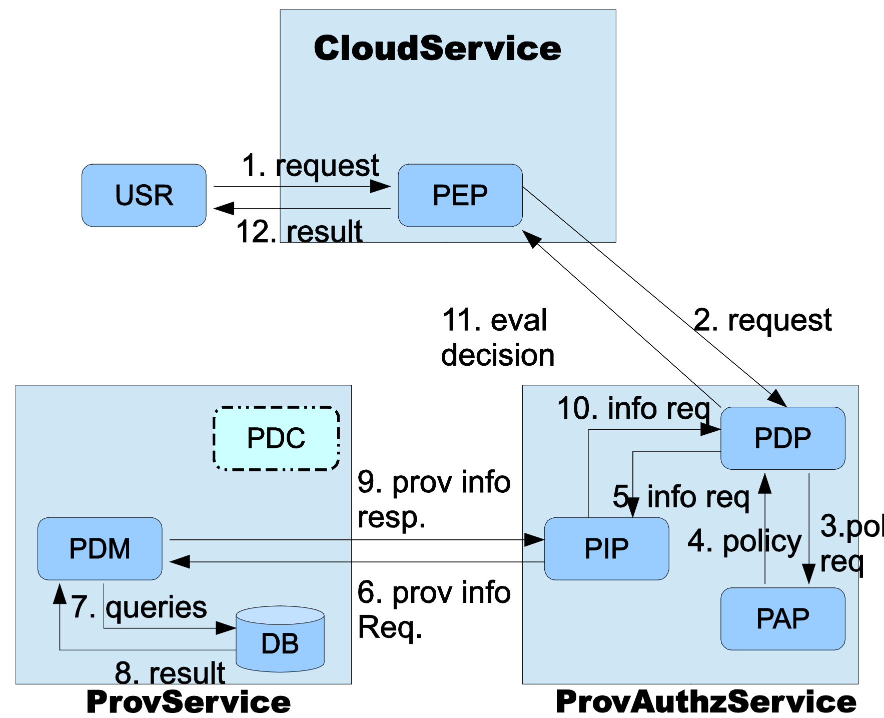
Provenance data sharing -

Logical Architecture

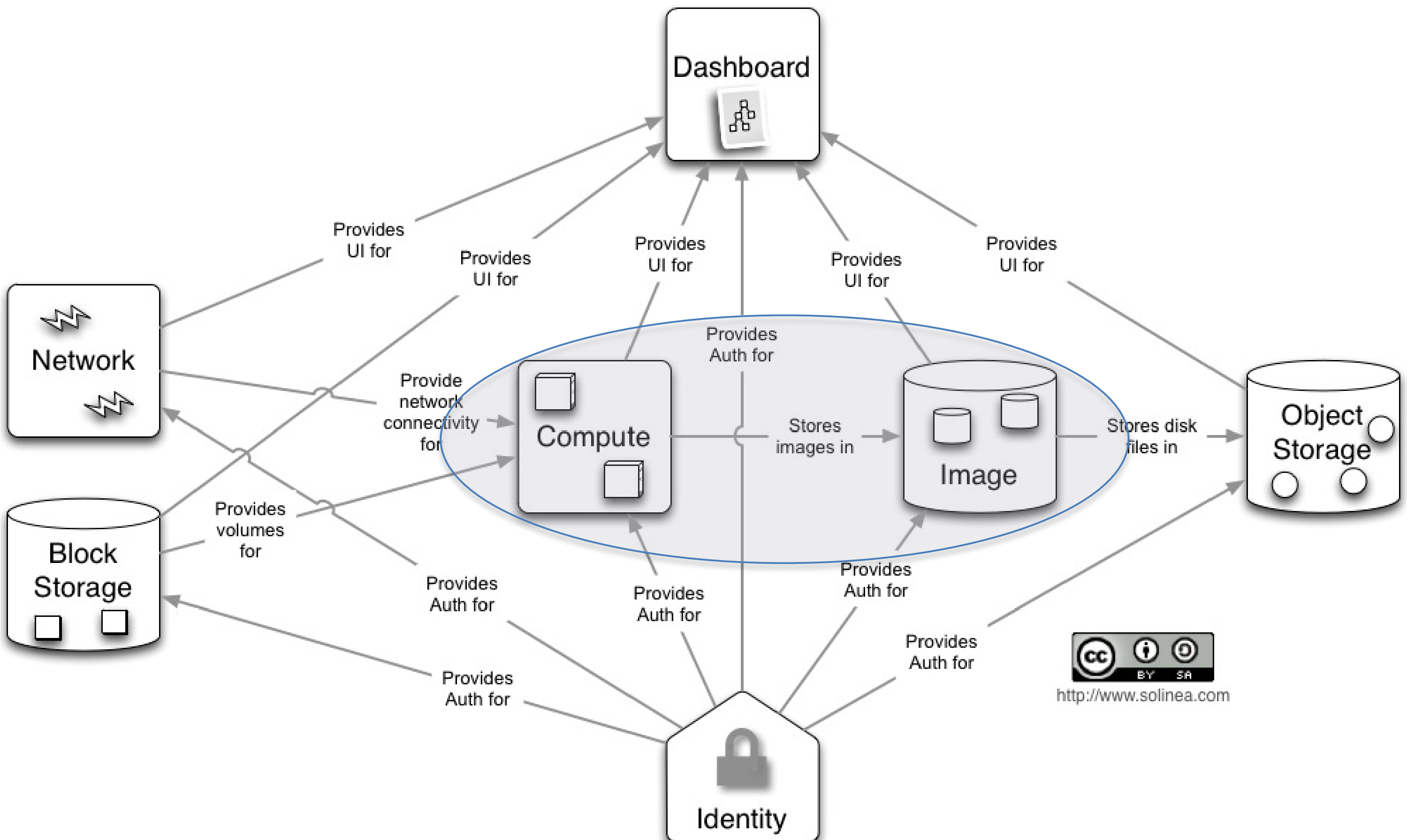


PROV-SERVICE Dataflow

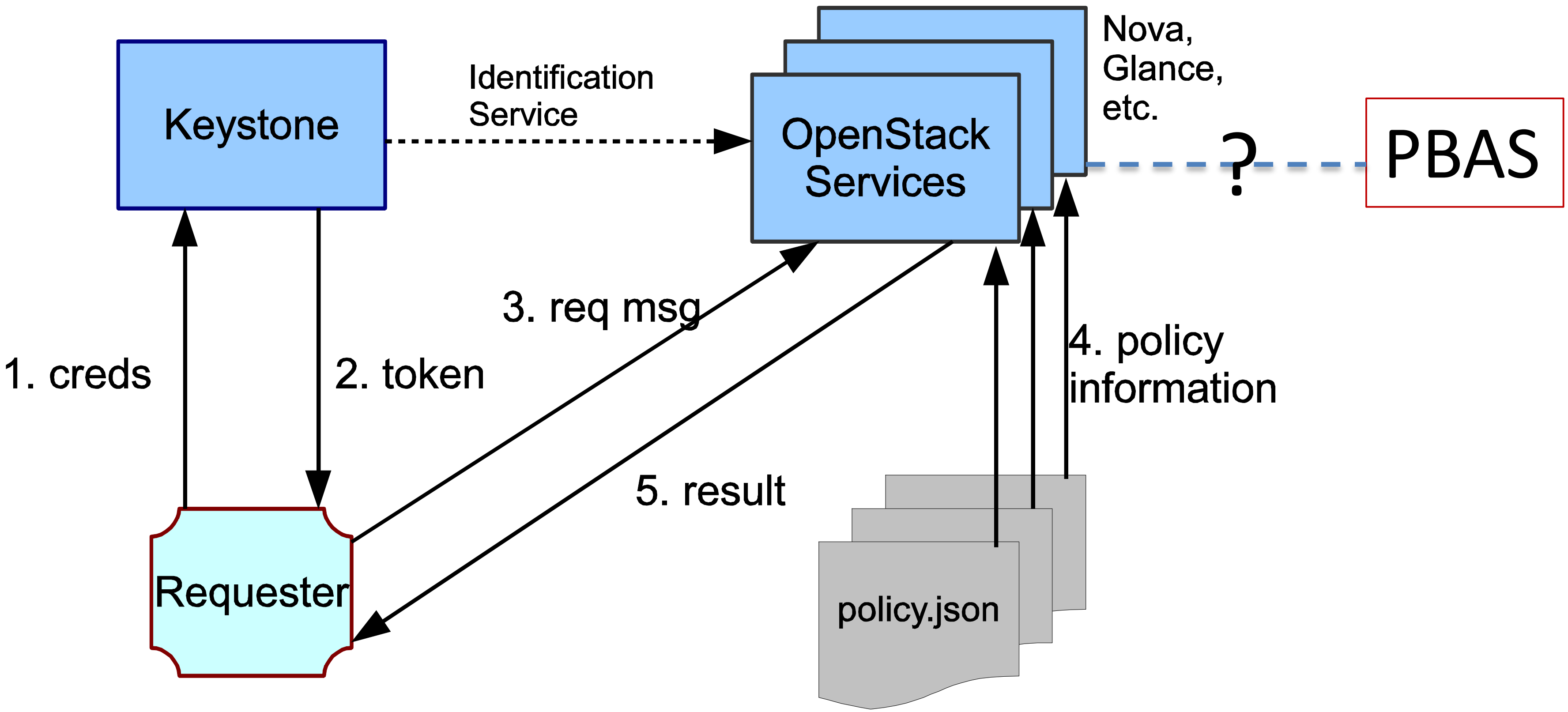
PROVAUTHZ-SERVICE Dataflow



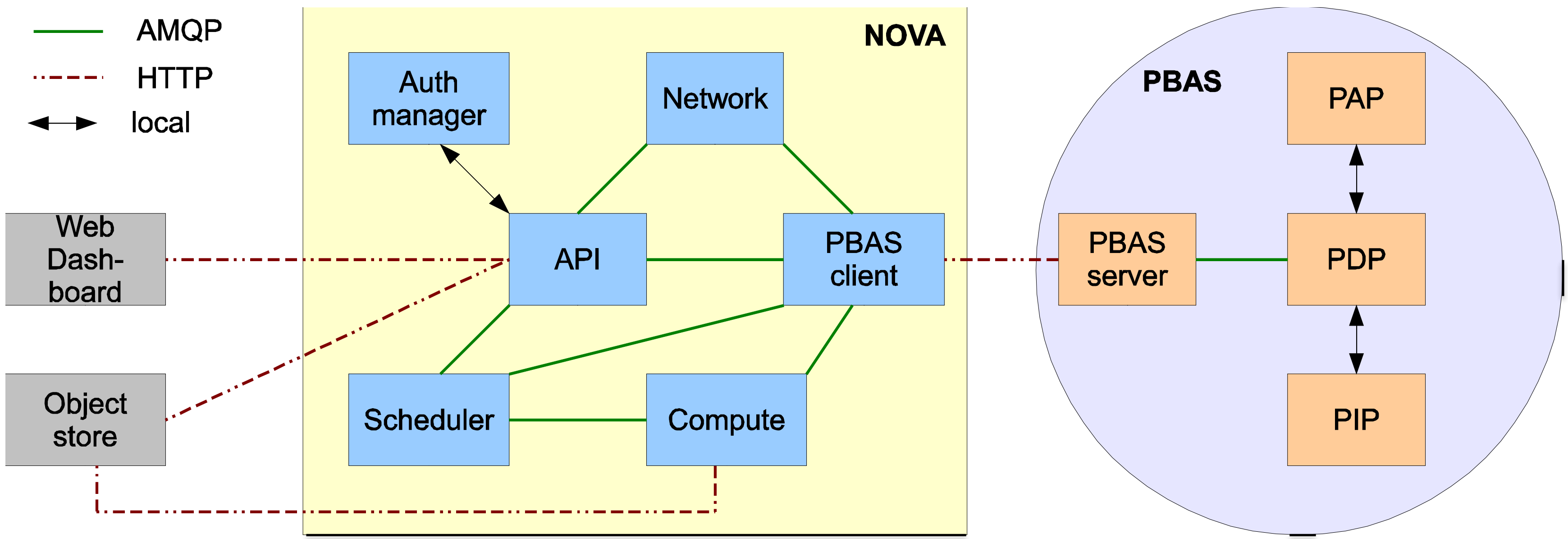
OpenStack Conceptual Architecture



OpenStack Authorization



Nova PBAS Implementation



Experiments

- Measure the time an authorization process takes from the time of request until decision is returned.
  - nova list
  - glance image-list
- 4 experimental configurations:
  - E1: normal Nova and Glance authorization.
  - E2: integrated PBAS/PS services with Nova and Glance.
  - E3: integrated PBAS/PS service, stand-alone from Nova and Glance.
  - E4: separate PBAS and PS services, stand-alone from Nova and Glance.
- Deployment Configurations:
  - 4GB RAM, 2.5 GHz quad-core CPU.
  - OpenStack Devstack (Grizzly) on 12.04 Ubuntu.
- Mainly test deep-shaped provenance graphs.
  - Generate mock data for virtual images and machines scenario.

Results and Evaluation

Traversal Distance	Glance (e1)	Glance (e2)	Glance (e3)	Glance (e4)
No PBAC	0.55	-	-	-
20 Edges	-	0.575	0.607	.642
1000 edges	-	.612	.788	.852

Traversal Distance	Nova (e1)	Nova (e2)	Nova (e3)	Nova (e4)
No PBAC	0.75	-	-	-
20 Edges	-	0.84	0.902	1.062
1000 edges	-	2.292	.362	4.102

Conclusion

- ✓ Proposed a framework of **provenance data and PBAC models** for enhanced access control.
- ✓ Proposed an **architecture** that enables PBAC and PS in cloud IaaS.
- ✓ Proof-of-concept **prototypes**
    1. XACML architecture extension and evaluation.
    2. OpenStack architecture extension and evaluation.
- ➢ An access control foundation for secure provenance-centric computing!

Future Work and Directions

- ❑ Expanding provenance data model to include **user-declared** provenance data.

- ❑ **Collaborated** PBAC usage
  - Multi-cloud.
  - Distributed systems.

- ❑ Full-cycle implementation and evaluation
  - including **provenance capturing** service.

- ❑ **Provenance Access Control** models and mechanisms.
  - Utilizing PBAC foundations.

Publications

1. **Dang Nguyen**, Jaehong Park and Ravi Sandhu, Adopting Provenance-Based Access Control in OpenStack Cloud IaaS. In Proceedings 8th International Conference on Network and System Security (NSS 2014), Xi'an, China, October 15-17, 2014, 15 pages.
2. **Dang Nguyen**, Jaehong Park and Ravi Sandhu, A Provenance-based Access Control Model for Dynamic Separation of Duties. In Proceedings 11th IEEE Conference on Privacy, Security and Trust (PST), Tarragona, Spain, July 10-12, 2013, 10 pages. (Best Student Paper Award)
3. **Dang Nguyen**, Jaehong Park and Ravi Sandhu, Integrated Provenance Data for Access Control in Group-Centric Collaboration. In Proceedings 13th IEEE Conference on Information Reuse and Integration (IRI), Las Vegas, Nevada, August 8-10, 2012, 8 pages.
4. Jaehong Park, **Dang Nguyen** and Ravi Sandhu, A Provenance-Based Access Control Model. In Proceedings 10th IEEE Conference on Privacy, Security and Trust (PST), Paris, France, July 16-18, 2012, 8 pages.
5. **Dang Nguyen**, Jaehong Park and Ravi Sandhu, Dependency Path Patterns as the Foundation of Access Control in Provenance-Aware Systems. In Proceedings 4th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2012), Boston, MA, June 14-15, 2012, 4 pages.
6. Jaehong Park, **Dang Nguyen** and Ravi Sandhu, On Data Provenance in Group-centric Secure Collaboration. In Proceedings 7th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Orlando, Florida, October 15-18, 2011, 10 pages.

Additional Publications

7. Lianshan Sun, Jaehong Park, **Dang Nguyen** and Ravi Sandhu. A Provenance-aware Access Control Framework with Typed Provenance. Pending revision for Transactions on Dependable and Secure Computing (TDSC), 2014.

8. Elisa Bertino, Gabriel Ghinita, Murat Kantarcioglu, **Dang Nguyen**, Jae Park, Ravi Sandhu, Salmin Sultana, Bhavani Thuraisingham, Shouhuai Xu. A roadmap for privacy-enhanced secure data provenance. Journal of Intelligent Information Systems, 2014.

9. Yuan Cheng, **Dang Nguyen**, Khalid Bijon, Ram Krishnan, Jaehong Park and Ravi Sandhu, Towards Provenance and Risk-Awareness in Social Computing. In Proceedings of the First ACM International Workshop on Secure and Resilient Architectures and Systems (SRAS '12), Minneapolis, Minnesota, September 19, 2012, pages 25-30.

Thank you!!!

Questions and Comments?