Software Quality and Infrastructure Protection for Diffuse Computing

Access Control and Policy Management

Speaker: John Mitchell (Stanford)
SPYCE Objective: Scalable Distributed Assurance

Develop fundamental understanding, models, algorithms, and network testbed, in order to reduce cost, improve performance, and provide higher reliability for networked operations across untrusted networks.

Incentives, Privacy, and Anonymity

Protocol Design and Analysis

Network Architecture

Trust Management
Traction

- **Problem**
  - Access policy: specification and enforcement

- **Approach**
  - Tractable subsets of first-order logic

- **Accomplishments**
  - Policy languages, algorithms, demo applications

- **Transitions to Industry**
  - Collaboration, start-up, students to industry, industrial visitor to Stanford

- **Future**
  - Interface w/commercial approaches: XrML, EPAL, P3P
  - Policy development environment: algorithms, tools

*Present activity: Cornell, Stanford*  
*Past contributors: Penn, Yale*
Distributed Access Control

Policy at site A may govern resources at site B

Protect distributed resources with distributed policy
Diffuse Computing Goals

- Flexible and scalable access control for loosely-coupled, diffuse systems
  - Resource sharing
    - coalitions, multi-centric collaborative systems
    - grid computing
  - Personal Privacy
    - Protect suborganization’s input to diffuse system
    - Example: patient’s hospital records
  - Organizational Privacy
    - Protect institution’s assets from suborganizations
    - Example: coalition assets from coalition partners
Decentralized Policy Example

EPub

Grants access to university students
Trusts universities to certify students
Trusts ABU to certify universities

Alice

Alice is a student

ABU

StateU

StateU is a university
Merging Policy Sets

♦ COTS information-sharing framework
  - Uses industry-standard policy $P_{\text{ind}}$
♦ Military wants to run COTS application,
  - Wants military policies $P_{\text{mil}}$
♦ Must detect and resolve conflicts
  - Integrate $P_{\text{ind}}$ and $P_{\text{mil}}$
Enterprise Access Control

Policy

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>When</th>
<th>Where</th>
<th>Why</th>
</tr>
</thead>
</table>

Joe can open *financials.xls* on his laptop using wired SSL

User
Right
Resource
Resource
Constraint
Who
What
When
Where
Policy Management Lifecycle

Plan

Analyze

Enforce

Measure

Improve
Core Issue

Policy Language and Deduction

♦ Specification
  - State policy succinctly and directly
  - Confident that policy captures intention

♦ Enforcement
  - Deduction, proof of compliance

♦ Manage policy lifecycle
  - Policy development tools
  - Safety and availability analysis
SPYCE Approach

♦ Tractable subsets of first-order logic
  - Datalog
    • If-then patterns without function symbols or negation
    • Standard database query formalism
  - Constraint Datalog
    • Constraints based on constraint domains
    • Hierarchies, intervals, discrete sets
  - Polarity-restricted FOL (“Lithium”)
    • Allows function symbols, negation
    • Supports modular combination of policies
General policy form

A policy statement has the form:

$$\forall x_1, \ldots, \forall x_m (\text{Condition} \Rightarrow (\neg) \text{Permitted}(\text{principal}, \text{privilege}))$$

where

- **Condition** is a conjunction of literals;
- **principal** is individual [HW] or group [LM];
- **privilege** can be action [HW] or group [LM].
Role-based Trust-management (RT)

- **RT₀**: Decentralized Roles
- **RT₁**: Parameterized Roles
  - **RT₁ᵀ**: for Separation of Duties
  - **RT₁Ĉ**: structured resources
- **RT₂**: Logical Objects
  - **RT₂Ĉ**: structured resources
- **RT₂ᵀ**: for Separation of Duties
- **RT₁ᴰ**: for Selective Use of Role memberships
  - **RT₁Ĉ**: structured resources

RTᵀ and RTᴰ can be used (either together or separately) with any of the five base languages: RT₀, RT₁, RT₂, RT₁Ĉ, and RT₂Ĉ.
Recall example

- **EPub**
  - Grants access to university students
  - Trusts universities to certify students
  - Trusts ABU to certify universities

- **Alice**
  - Alice is a student

- **ABU**

- **StateU**
  - StateU is a university
Example $RT_0$ credentials

1. StateU.stuID ← Alice
2. ABU.accredited ← StateU
3. EPub.university ← ABU.accredited
5. EPub.access ← EPub.student

Together,
five statements prove Alice is entitled to access
Policy Diffusion

EPub

EPub.university ← ABU.accredited
EPub.student ← EPub.university.stuID

ABU

ABU.accredited ← StateU

Alice

COE.stuID ← Alice

StateU

StateU.stuID ← COE.stuID

COE
Natural
- Security policy statements are if-then rules

Precise
- Declarative and widely-understood semantics

Tractable
- No function symbols $\Rightarrow$ tractability
- Efficient goal-directed evaluation procedures

Available technology
- Extensive Datalog research in LP and DB
Better: Constraint Datalog

♦ Why constraints:
  - Datalog cannot easily express permissions about structured resources and ranges

♦ What is Constraint Datalog
  - Special form of CLP; query language for Constraint DB

♦ A Constraint Datalog rule:
  - $R_0(x_0) : - R_1(x_1), \ldots, R_n(x_n), \varphi(x_0, x_1, \ldots, x_n)$
    - $x_0, x_1, \ldots, x_n$ are tuples of variables
    - $\varphi$ is a constraint in all the variables
Example Policy with Constraints

♦ A grants to B the permission to
  - connect to hosts in the domain “stanford.edu”
  - at port 80,
  - valid from time \( t_1 \) to \( t_3 \), and
  - allows B to further delegate

\[
\text{grantConnect}(A, B, h, p, v) :\neg
\]

\[
h \subset \langle \text{edu}, \text{stanford} \rangle, \ p=80, \ v \in [t_1, t_3]
\]

\[
\text{grantConnect}(A, x, h, p, v) :\neg
\]

\[
\text{grantConnect}(B, x, h, p, v),
\]

\[
h \subset \langle \text{edu}, \text{stanford} \rangle, \ p=80, \ v \in [t_1, t_3]
\]
Useful Constraint Domains

♦ Tree domains:
  - Path expressions \( \langle a_1, a_2, \ldots, a_k \rangle \)
    - E.g., \( \langle \text{pub}, \text{software} \rangle \) for /pub/software
  - Primitive constraint: \( x = y \) or \( x \theta \langle a_1, \ldots, a_k \rangle \), where \( \theta \in \{ =, <, \leq, \subset, \subseteq \} \)

♦ Range domains:
  - Primitives: \( x = y \), \( x = c \), or, \( x \in (c_1, c_2) \)

♦ Discrete domains with finite sets:
  - Primitive constraint: \( x = y \), \( x \in \{ c_1, c_2, \ldots, c_j \} \)
RT Accomplishments

♦ RT language design
♦ RT deduction, trust negotiation
  - Fast distributed deduction, constraints
♦ Comparative analysis: KeyNote, SPKI 2.0
♦ Safety and availability analysis
  - HRU undecidability => RT poly-time, NP, Pspace
♦ Sample applications
  - August scheduling, UStorIt file sharing, ATN, ...
♦ Transitions
  - IBM privacy, InnerPresence, NTT DoCoMo, Hitachi
Complementary Approach

♦ RT starts with a tractable fragment of first-order logic (Datalog) and extends it
♦ [HW] start with an intractable language (first-order logic) and restricts it

Result: [HW] can capture policies that forbid actions expressed using functions
Sample policies

♦ Many policies explicitly forbid actions
  - `Smoking is prohibited in the dining areas of all restaurants seating more than 35 people’
    NYC Smoke-Free Air Act
  - `The tickets may not be refunded’ is a policy of many theaters, special airline fares, ...

♦ Functions are also useful
  \[\forall x_1 (\text{Absent}(x_1) \Rightarrow \text{Permitted}(\text{Spouse}(x_1), \text{SpeakFor}(x_1)))\]

♦ Expressive: [HW] not easily expressed in RT

♦ Tractable: [HW] fragment decidable in poly time
Multiple Policy Sets

Because [HW] supports prohibitions, can detect conflicts between policies

E.g: detect conflicts between $P_{\text{ind}}$ and $P_{\text{mil}}$
- Find actions permitted, forbidden in $P_{\text{ind}} \cup P_{\text{mil}}$
- Requires language that express prohibitions
- If the policy language assumes any action that is not explicitly permitted is forbidden, then $P_{\text{ind}}$ and $P_{\text{mil}}$ conflict, unless they’re identical
Policy Combination

Denied + Denied = Denied

Denied + Permitted = Permitted

Denied + Permitted = Denied

Denied + Denied = ???

Denied + Permitted = Permitted
A Formal Foundation for XrML

♦ XrML: a language for writing software licenses.
  - Very popular: to be supported by Microsoft, +++
♦ Specification gives syntax, informal semantics, + algorithm to determine if a permission follows from a set of licenses.
♦ What we’ve done:
  - Provided formal semantics for XrML
  - Found bugs in their algorithm
  - Proved their queries are NP-hard and
  - Found tractable fragments of XrML.
♦ Interacting with XrML standards committee

See Vicky’s poster for more details!
Critical Infrastructure Protection

Many critical infrastructures, national and DoD-specific, are decentralized.

Data sharing essential for operation, but data compromise can be catastrophic.

Research Question: How to share data safely, using policies that are easy to formulate, enforce, maintain.

Approach: diffuse trust management.
Assuring Software Quality

♦ Technology applicable to managing process interaction
  - Process A delegates rights to process B
    • For limited purpose, limited time, limited locations
  - Fine-grained control of process actions
  - Works for diffuse systems that escape normal controls imposed by localized OSs

♦ Diffuse principle of least privilege
DoD Impact

♦ Dynamic coalitions
  - Partial sharing based on partial trust

♦ Joint Vision 2010 / Joint Vision 2020 of “Network Centric” operations
  - Can use policy to push data, overcome network bandwidth limitations
  - Right data to right place at right time
Plans for Option

♦ Applications and Transitions
  - Work with XrML developers on language and algorithm
  - IBM Privacy Project
    • Use RT algorithms for EPAL, P3P applications
  - Pursue commercial and DOD applications
  - Application to large policy sets (social security policies)

♦ Generalize results: RT ⇒ Datalog ⇒ PFOL

♦ Improve implementation: RT₀ ⇒ Datalog ⇒ PFOL

♦ Policy development environment and tools
  - User interface, XML-format, interoperability
  - Testing methodology, analysis methods
Architecture design for policy-based portal site

**Portal site**

- **User**
  - doesn’t have a federated ID
  - has a federated ID

- **Bob’s Policy(A)**
  - Bob.Address <-> CA
  - Bob.Job <-> Student

- **General Policy**
  - Discount <-> Student

- **Web service**
  - invoke service
  - return results (customized by General Policy)

- **Attribute Authority(A)**
  - (SAML) Attribute Assertion

- **Authentication Authority**
  - (SOAP)
  - discover
  - register
  - (UDDI) register

- **UDDI Registry**
  - discover
  - invoke service
  - return results (customized by Bob’s Policy(B))

- **Attribute Authority(A)**
  - (Liberty Alliance) federated ID

- **Attribute Authority(B)**
  - (Liberty Alliance) federated ID

- **Policy Decision Authority**
  - create

- **Service Provider**

**Credential Chains**

- **Bob’s Policy(A)**
  - Bob.Address <-> CA
  - Bob.Job <-> Student

- **Bob’s Policy(B)**
  - Bob.MembershipPoint <-> high

**Role-based Trust Management (TM) language**

**Language**

- (SOAP)
- (SAML)
- UDDI

**Hitachi**
Secure service infrastructure for heterogeneous overlay networks

“Diffusion”

- Communication
- Cooperation
- Incentives
- Delivery

Secure service infrastructure for heterogeneous overlay networks
Automated Trust Negotiation

- Credentials may contain sensitive information
  - need protection just as other resources
  - deduction must be interactive

- The Trust Target Graph (TTG) protocol
  - supports RT₀, which has delegation
  - supports distributed discovery of statements
  - supports Ack policies, which also protects against unauthorized leakage of attribute information
Safety and Availability Analysis

♦ Organizations delegate partial control
  - What happens if other organizations change policy in the future without my knowledge?

♦ Given policy $P$ and restriction $R$ on changes
  - Simple safety: Is $A \cdot r \supseteq \{D\}$ possible?  $\Rightarrow$ PTIME
  - Simple availability: Is $A \cdot r \supseteq \{D\}$ necessary?  $\Rightarrow$ PTIME
  - Bounded safety: Is $\{D_1, \ldots, D_n\} \supseteq A \cdot r$ necessary?  $\Rightarrow$ PTIME
Complexity of Containment Analysis

Given \( P \) and \( R \), is \( A.r \supseteq B.r1 \) necessary?

- Simple delegation \( \Rightarrow PTIME \)
  - Uses logic programs with stratified negation
- Intersection \( \Rightarrow coNP\)-complete
  - Equivalent to determining validity in propositional logic
- Linking \( \Rightarrow PSPACE\)-complete
  - Equivalent to containment of languages accepted by NFA

Decidability, PTIME stand in contrast to the HRU model, in which simple safety is undecidable

- Linking + Intersection \( \Rightarrow \) decidable in coNEXP
  - Exact complexity unknown
Implementation Status

♦ Java inference engine for RT\(_0\)
♦ Preliminary version of RTML
  - an XML-based Encoding of RT statements
  - XML Schemas and parser exist
♦ Applications
  - U-STOR-IT: Web-based file storage and sharing
  - August: A Distributed Calendar Program
  - Automated Trust Negotiation Demo by NAI
  - TNT Trust Negotiation architecture at BYU
NEED TO UPDATE
Future SPYCE Directions

♦ Accomplishments
  - Framework and logic for policy definition
  - Algorithms for policy enforcement
  - Some experience with capturing practical policy requirements from a variety of applications

♦ Challenges
  - Continue implementation and deployment efforts
  - Policy development algorithms and tools
  - Debugging and testing safety and availability
Sample Applications

♦ August Distributed Calendar
♦ USTORIT
♦ Social security database
  - policy to qualify for social security
♦ Library policy
  - Have to administer copyright
  - Who is allowed to access course notes?
♦ XrML
  - Commercial license and rights framework
August: Distributed Calendar

♦ Users define groups, maybe interdependent
♦ Each user has a calendar and can specify policy which determines
  – who is allowed to view each part of the user's calendar
  – who is allowed to add an activity of a certain kinds at a certain time
U-STOR-IT: Web-based sharing

♦ User auth: SSL and X.509 certificates
♦ Users define interdependent groups
♦ Locker
  - Hierarchical folder of heterogeneous files
  - Locker/file upload/access policy set by owner
♦ Policies translated into RT
♦ Implement w/Java Servlets and MySQL
IBM collaboration, Startup