Integrity Constraints in Trust Management

Sandro Etalle
University of Twente
now (2007) visiting the University of Trento

William H. Winsborough
University of Texas S. Antonio
Plan of the talk

- Trust Management in a Nutshell
  - Rule based vs reputation based
  - Issues & Research Challenges

- A solution to a specific problem
  - Integrity Constraints

- Why should we care
  - Applications in Research Projects
Part 1: Trust Management in a Nutshell

Two flavours: reputation based & rule based

Issues and research challenges
The other TM: “Reputation Based”

- **Concrete:**
  - community of cooks (200 people)
  - need to interact with someone you don’t know,
  - to establish trust you ask your friends
    - and friends of friends
  - Dynamic: after success trust increases

- **Virtual:**
  - p2p community of hackers (5000 people)
  - Larger (scalability)
  - Trust establishment has to be automatic
rule-based TM: a concrete example

- Bart is entitled to a discount

*If he is a student of the local university*
When is Bart now entitled to a discount?
Bart is now entitled to a discount …

- If he is a student of any accredited University.
- But perhaps there are other reasons why Bart is entitled to a discount
  - If he is an employee of any governmental organization
  - If he is a member of the library club
  - If he is a veteran
  - ....

- Problems
  - Scalability
  - Knowing where and what to search
Issue 1: Credential Chain Discovery

- Different security domains (explains Trust).
- We have a chain of credentials (distributed)
- Issue 1: finding the chain
Issue 2: Trust Negotiation

- No predefined security monitor
- Credentials need to be disclosed to a possibly untrusted party
- Credentials may contain private info (e.g. medical record)

**Issue 2**: Trust negotiation

- Goal: establish trust while maintaining privacy
  - by iterative disclosure of credential
Issue 3: expressiveness

- Decentralized + negotiation
  - if you can’t find a credential it does not mean it is not there
  - additional disclosures should not lead to revocations
  - monotonicity?
- “a buyer may also be an accountant, provided that her behaviour is logged”
- How do we deal with this?

ISSUE 3: expressiveness
Solutions to Issue 3 (our research)

- In the language
  - Allow a tamed version of negation
    - Negated atoms are guarded & safe
    - Paper [Cze05], roots in [Eta96].
- Outside the language
  - Use Integrity Constraints
Part II
Integrity Constraints in TM

- Quick intro RT language
- Integrity Constraints
Role-based Trust Management ($RT$)

- A family of credential/policy languages
  - Simplest, $RT_0$, has no parameterization, thresholds, or separation of duty [Li, Mitchell, Winsborough]

$RT_0$ example: student discount subscription

- $EPub\.studentDiscount \leftarrow StateU\.student$
- $StateU\.student \leftarrow URegistrar\.fulltimeLoad$
- $StateU\.student \leftarrow URegistrar\.parttimeLoad$
- $URegistrar\.parttimeLoad \leftarrow Alice$
**RT₀ Syntax**

- **A, B, D:** principals
- **r, r₁, r₂:** role names
- **A.r:** a role (a principal + a role name)

**Four types of credentials:**

- **A.r ← D**  
  Role A.r contains principal D as a member
- **A.r ← B.r₁**  
  A.r contains role B.r₁ as a subset
- **A.r ← A.r₁.r₂**  
  A.r ⊇ B.r₂ for each B in A.r₁
- **A.r ← A₁.r₁ ∩ A₂.r₂**  
  A.r contains the intersection

- The first 3 statement types: equivalent to pure SDSI
- More complex versions have parameters (RT₁), constraints (RTₐ), and can model thresholds and separation of duty (RTₜ)
Example: Attribute-based Delegation

- Accepting student ID from any university

  - Credential chain proves authorization.
  - Declarative semantics: \([[[EPub\text{.}discount]]_P = \{ Alice \}]\)
Why Integrity Constraints

- Policies do change: \( P \Rightarrow P_1 \Rightarrow \ldots \Rightarrow P_n \)

- A principal controls only a portion of the policy
  - Statements may be added or removed by other principals
  - Nowadays: trusted principals give no feedback to the trusting ones

- Delegating trust implies an understanding between principals,
  - Nowadays: not formalized

- Trusted principals need assistance in understanding global impact of delegations, revocations
  - Who could get access to what? (Safety)
    - Assessing exposure
  - Who could be denied? (Availability)
    - Ensuring applications have authorizations needed for correct operation
Problem Instances

- Mutual Exclusion
  - “No-one should ever be both a buyer and an accountant”

- Containment
  - “Welders of BOVAG-accredited workshops should be fellows of the British Institute of Welding”

- Can’t express this in the language.
Integrity Constraints: General Form

- **General:** $L.l \sqsupseteq R.r$
  - $L.l \sqsupseteq R.r$ holds in $P$ iff $[[L.l]]_P \sqsupseteq [[R.r]]_P$
  - $L.l$ and $R.r$ may be sets and intersections of roles

- **Special cases**
  - Membership: $A.r \sqsupseteq \{D_1, \ldots, D_n\}$
  - Boundedness: $\{D_1, \ldots, D_n\} \sqsupseteq A.r$

  - expressiveness is limited (it is a universal formula) but we can express all safety properties of [LWM03]
  - counterexample: at least a manager should have access to the DB
Examples

- **Buyers and accountants should be disjoint**
  - $\emptyset \sqsubseteq A.\text{buyer} \cap A.\text{accountant}$

- **Welders of BOVAG-accredited workshops should be fellows of the British Institute of Welding**
  - Bovag.welder $\leftarrow$ Bovag.accr.welder
  - Bovag.accr $\leftarrow$ PietersWorkshop
  - PietersWorkshop.welder $\leftarrow$ Pieter
  - BIW.fellow $\sqsupseteq$ Bovag.welder
The technical problem

- P ⇒ P1 ⇒ ... ⇒ Pn: policy change
- L.l ⊒ R.r: a constraint

Need a (minimal) mechanism such that
- IF L.l ⊒ R.r does not hold in Pi
- THEN a warning is fired
- without checking L.l ⊒ R.r each time a credential is added/removed

How: by monitoring when some credentials are added or removed
The solution in short

- **P** – Policy,
- **Q** = **L.I** ⊆ **R.r** – Constraint

Let **G** = roles **R.r** *depends on*
- \( \forall X \ [ [R.r]_P = [ [R.r]_{P \cup (X \setminus G)}] \)

Prop: can compute a “small” **G** in \(|P|^2\)

Let **S** = set of credentials s.t.
- \( [ [L.I]_{P \cup S} \supset [ [R.r]_P] \)

**Theorem:**

- Let \( P \Rightarrow P1 \Rightarrow ... \Rightarrow Pn \)
- IF
  - \( P \) satisfies **Q**
  - no credential **S** is removed
  - no credential for **G** is added
- THEN
  - \( Pn \) satisfies **Q**
  - **G** and **S** don’t have to be recomputed
The method

- **P** – policy,
- **Q** = **L.I** ⊑ **R.r**: constraint

**CHECKING**
- FIRST, compute **[[R.r]]_P**
  - here **G** is computed “for free”
- THEN, for each **X ∈ [[R.r]]_P**, check that **X ∈ [[L.I]]**
  - here (one of the) **S** is computed “for free”

**MONITORING**
- Let **P ⇒ P1 ⇒ ... ⇒ Pn**
- IF
  - no credential for **S** is removed
  - no credential for **G** is added
- Then
  - OK
- Otherwise
  - Check **Q** again, and
  - Recompute **G** and **S**
    - (even if **Q** still holds)

To monitor this we need the cooperation of other principals
Extra Difficulty: non cooperating principals

- $P \Rightarrow P_1 \Rightarrow \ldots \Rightarrow P_n$: policy change
- $L.l \sqsupseteq R.r$: a constraint
- UT: set of untrusted principals
- $P'$ is reachable from $P$ iff $\text{diff}(P,P') \subseteq \text{roles in UT}$

Need a (minimal) mechanism such that
- IF $L.l \sqsupseteq R.r$ does not hold in some $P'$ reachable from $P_i$
- THEN a warning is fired
- without checking $L.l \sqsupseteq R.r$ each time
The method with non cooperating principals

- P – policy,
- Q = L.I ⊆ R.r: constraint
- T = trusted roles

Define
- UB(P) and LB(P): new semantics [LMW04]
- G = ... see paper ...
- S = credentials such that
  - \([L.I]_{LB(P|S)} \supseteq [R.r]_{UB(P)}\)

Theorem:
- Let \(P \Rightarrow P_1 \Rightarrow \ldots \Rightarrow P_n\)
- IF
  - \([L.I]_{LB(P)} \supseteq [R.r]_{UB(P)}\) and
  - no credential S is removed
  - no credential for G is added
- THEN
  - Every \(P'\) reachable from \(P_n\) satisfies Q
  - G and S don’t have to be recomputed
Future work

- P ⇒ P₁ ⇒ ... ⇒ Pₙ – policy change,
- Q = L.l ⊑ R.r – integrity constraint

Challenge:
- Isolate a set S of roles
- For each X in S derive from Q a constraint Qₓ “local to X” s.t. Q is violated only if for some X, Qₓ is violated

Problem
- cyclic dependencies
Conclusions

- **Integrity constraints:**
  - tool to control a TM system.

- **Monitoring requires the cooperation of trusted principals**

- **Trust management becomes a two way process**
  - from the trusting to the trusted
  - and vice-versa
Part III – why should we care

2 Examples
Application Areas

- Business to Business
  - X-organization cooperation
- Mobile (roaming) Devices
- P2P

- Combines well with **sticky policies** [ew07]
  - E-health
  - ...

- Anywhere where there are various security domains
Questions?
References to own work