Trust and Security Policy Specification for Internet Applications

Morris Sloman

Department of Computing
Imperial College
m.sloman@doc.ic.ac.uk,
http://www.doc.ic.ac.uk/~mss

Acknowledgment

Others contributing to this work:
- Nicodemos Damianou
- Naranker Dulay
- Tyrone Grandison
- Emil Lupu

Contents

Trust specification and analysis
Ponder policy specification language
Basic policies – authorisation, refrain, obligation, delegation
Composite policies – roles, relationships, management structures
Conflicts
Conclusions and future work

What is Trust

- A quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context

- Dependability implies timeliness
- Distrust useful for trust revocation or in default trusted environments
- Quantification various degrees of trust/distrust
Trust Classification

- Provision of service by trustee eg financial advice
- Certification of trustee eg BMA or Verisign
- Delegation of trust eg accountant makes all my investment decision
  Although trust is not usually transitive
- Infrastructure trust eg trusted computer system and network

Trust Specification

- Trust Predicate
  trust (trustor, trustee, actions, level, ) ← constraint set
  trust (Helen, _doctor, heart_diagnosis; operate, 50) ←
  is_consultant (_doctor, NHLI)
- Distrust when level < 0
- Recommend Predicate
  recommend (recommedor, recomendee, actions, level) ← constraint set
  recommend (Morris, J.Bloggs, WebProgram, high) ←
  has_degree (IC-computing, 2i)
  trust (Harry, Frank, DesignHouse, medium) ←
  recommend (Tom, Frank, DesignHouse, high)

Trust, Experience and Risk

- Trust is not static but changes with time as a result of experience
- High risk low trust
- Trust framework must monitor experience, risk and constraints in order to dynamically update trust levels and relationships.

Trust Analysis

- Determine trust relationships eg
  - What are the relationships between A & B?
  - Who trusts B?
  - What relationships have A as recommendor?
  - Conflict of interest relationships
- Determine implicit trust relationships
- Generate security management policies from trust specification
Trust specification and analysis

**Ponder policy specification language**

Basic policies – authorisation, refrain, obligation, delegation
Composite policies – roles, relationships, management structures
Conflicts
Conclusions and future work

---

Security Specification

- E-commerce, healthcare – multiple organisations
- Complex security policies with many constraints and exceptions
- Implementation is often the specification
- Need to specify security policy for groups and roles (organisational positions)
- Need to manage security – what actions to perform when a violation detected, or for registering a user
- Need for analysis tools

---

Policy

Rule governing choices in behaviour of the system

- Derived from trust specifications, enterprise goals and service level agreements
- Need to specify and modify policies without coding into automated agents
- Policies are **persistent**
- But can be dynamically modified
  Change system behaviour without modifying implementation – **not new functionality**

---

Ponder Policy Language

- Precise specification of subjects, targets, actions and constraints for authorisations and obligations
- Needed for both:
  - **Human managers**
    - Clear specification of responsibility, rights and duties “job description”
  - **Automated agents**

---
Domains

- A domain is a collection of objects which have been explicitly grouped together for management purposes e.g. to apply a common policy (LDAP) directory
- Specify policy for groups of objects
- Can change domain membership without changing policy

Policy Propagation

Authorisation Policy

- Defines what a subject is permitted or not permitted (prohibited) to do to a target
- Protect target objects from unauthorised actions
  - Target based interpretation and enforcement

Authorisation Policies

- All policies can be specified as a parameterised type from which instances can be created

```java
type auth+ conf (subject s, target t, string start, string end) ;
  action VideoConf ();
  when time.between (start, end); }

inst marketConf = conf (/UK/marketing, /US/marketing “1400”,“1900”);
planConf = conf (/UK/planning, /France/management “0900”, “1200”);
```
Negative Authorisation

- Used for revocation of access rights

```plaintext
inst auth- revoke {
    subject  /users/JoeBloggs;
    target   /resources/database ;
    action   - ; // any action
    when     time.date > 30:9:2001 }
```

- Reflect organisational policies and laws

```plaintext
inst auth- nostrangle {
    subject  staff;
    action strangle;
    target  students; }
```

Refrain Policy

```plaintext
inst refrain politeBehaviour {
    subject Agroup ;
    target  AGroupNY + DGroupBoston ;
    action videoconf ;
    when   (time.day=Friday); }
```

- Similar to negative authorisation but subject based interpretation

Filters

- Transformations on parameters of positive authorisation policies, where it is not practical to provide different operations to reflect permitted parameters

```plaintext
inst auth+ employeeAccess {
    subject  employees + managers ;
    target   <DB> employeeDB ;
    action getEmp (empID) ;
    if (subject = employees)
        result = reject (result, salary, homeAddr); }
```

Obligation Policy

- Defines what actions a subject must do
- Assumes well behaved subjects with no freedom of choice.
- Subject based subject interprets policy and performs actions on targets
- Event triggered obligation
- Actions can be remote invocations or local scripts
- Can specify sequencing or concurrency of actions
Obligation Example

- After 3 consecutive login failures with the same userid, disable the userid, notify the administrator & log the userid.

```java
// LoginFailure action

type oblig LoginFailure (subject s, set b, string phoneNo) {
  on LoginFail (userID) {
    target <userT> t = b ^ {userID} ;
    do t.disable(userID ) -> ( s.sms(phoneNo, userID, "message....")
     || s.log (userID) );
  }
}

inst p = LoginFailure (/Nregion/secAgent, /Nregion/users, "07710123456") ;
```

Delegation Policy

- Specify which actions a subject can delegate to a grantees
- Must be a subset of subjects, actions and targets in an authorisation policy

```
// Delegation Policy
```

Contents

- Trust specification and analysis
- Ponder policy specification language
- Basic policies – authorisation, refrain, obligation, delegation
- Composite policies – roles, relationships, management structures
- Conflicts
- Conclusions & future work

User Representation Domain

- Persistent representation of a registered user
- URD is subject of policies applying to a specific person
- At login adapter object created to represent and act on behalf of person in system command interpreter
Roles

- Role groups the rights and duties related to a position in an organisation
- E.g., network operator, network manager, finance director, ward-nurse
- Specify policy in terms of roles rather than persons
do not have to re-specify policies when person assigned to new role

Role Example

type role op (target firewalls) {
  // load new firewall policies
  inst oblig intruder {
    on intrusion ( );
    do firewalls.load (highSec) ;
  }  

  inst auth+ fwAuth {
    target firewalls;
    action load, unload, readlog;
  }

  // other authorisation and obligation policies
}

Role Instances

- Multiple operator role instances
- Different persons assigned to roles
- Different target components
- Similar policies
- Role Class
- Reuse of role specification
Role Specialisation

- Derive new composite policy specifications from existing ones
- Specialise roles by adding policies

Inheritance

Operator

Router Operator

Network Administrator

Role Relationships

- Relationships
  - Rights and duties of roles towards each other
  - Usage of shared resources
  - Interaction protocols

Network Administrator

Management Structures

- Configurations of roles and relationships in organisational units

Ward Management Structure

```cpp
type mstruct ward (domain patients) {
    import /type/wnurseT; /type/hnurseT;

    inst role wardHead = hnurseT;
    wardNurse = wnurseT;
    rel supervise (wardHead, wardNurse) {
        inst auth+ {subject wardHead; target wardNurse; action assign () ;
        }
        inst oblig ..........;
    }
}
```
Organisational Patterns

Patient Records

\[ \text{inst mstruct} \]
\[ \text{hospital/ward1 = ward(w1Patients);} \]
\[ \text{hospital/ward2 = ward(w2Patients);} \]

Organisational Patterns

Patient Records

\[ \text{ward1} \]
\[ \text{wardHead} \]
\[ \text{supervise} \]
\[ \text{Policies} \]
\[ \text{Nurse} \]

\[ \text{ward2} \]
\[ \text{wardHead} \]
\[ \text{supervise} \]
\[ \text{Policies} \]
\[ \text{Nurse} \]

Contents

- Trust specification and analysis
- Ponder policy specification language
- Basic policies – authorisation, refrain, obligation, delegation
- Composite policies – roles, relationships, management structures
- Conflicts
- Conclusions & future work

Multiple Policies May Apply

- An object can be a member of multiple domains (overlap)
- Multiple policies can apply to single domain
- Need conflict detection and resolution

Modality Conflicts

- Potential conflict from overlap of subjects, targets and actions
- 3 types: auth+/auth-, oblig/auth-, oblig/refrain
- Note: auth+/refrain is not a conflict
- Detected by syntactic analysis
Example Conflicts

- \text{inst} auth- bootWS \{ 
  \text{subject} students; \text{target} workstations; \text{action} reboot \}
- Exception:
  \text{inst} auth+ projectWS \{ 
  \text{subject} smith; \text{target} workstations/project; \text{action} 
  reboot \}

Precedence

- Can resolve some conflicts automatically by specifying precedence, e.g.:
  - \textbf{Negative} policies override
    Does not permit positive exceptions to negative policies.
  - More \textbf{specific} policies override
  - Explicit priority

Semantic Conflicts

- Types of conflict:
  - \textbf{separation of duty} e.g., the same person is not allowed to authorise payments and initiate them
  - \textbf{self-management} e.g., a manager cannot authorise its own expenses
  - \textbf{conflict for resources} e.g., not more than 5 persons are authorised to change the DB
- Need to specify the conditions which result in conflict
- Constraints on a set of policies (Meta-Policies). Specified using Prolog, OCL
- Included in composite policies such as roles or mstructs

Separation of Duties

/policies/accounting->exists (P1, P2 | 
  P1.subjects->intersection(P2.subjects)->notEmpty and 
  P1.actions->exists(a | a.name = 'authorise') and 
  P2.actions->exists(a | a.name = 'initiate') and 
  P1.targets->intersection(P2.targets)->exists(t | 
    t.isOclKindOf(payment)))
Contents

- Trust specification and analysis
- Ponder policy specification language
- Basic policies – authorisation, refrain, obligation, delegation
- Composite policies – roles, relationships, management structures
- Conflicts

Conclusions and future work

Conclusions

- Security specification
- Management
- Large scale
- Multiple Organisations

- Authorisation, delegation, role
- Event-triggered obligation
- Domains + Composite policies

Future Work

- Refinement and analysis tools – Requirements Engineering approach
- Refinement of Trust into security policy
- Adaptive Security Management
- Service Level Agreements
- Authorisations + obligation policy for dynamic aspects
- Policy based programmable networks
  3 levels: applications, routers, hardware

Policy Workshop

- Policy 2002: Workshop on Policies for Distributed Systems and Networks
  http://www-dse.doc.ic.ac.uk/Events/policy-2002/
  5-7 June 2002
  Naval Postgraduate School
  Monterey, Ca, USA.

- Colocated with
  SACMAT 2002: 7th ACM Symposium on Access Control Models and Technologies
  3-4 June 2002, NPS, Monterey.
Additional Information

- Links to Policy information
  - Ponder
  - Workshops
  - Papers
  http://www-dse.doc.ic.ac.uk/policies