Trust Negotiation on the Grid

J. Basney  NCSA
W. Nejdl    L3S
D. Olmedilla  L3S
V. Welch    NCSA
M. Winslett  University of Illinois
Outline

- Introduction
- Trust Negotiation
- PeerTrust Language
- How PeerTrust Works
- Conclusions & Further Work
Traditional Access Control for Decentralized Systems

Assumption: I already know you---you have a local account!

Not a member?

Grid just replaces the login-password with an identity/proxy certificate
Motivating Example

MyProxy Credential Repository

0a Request previously stored proxy certificate

0b Receive proxy certificate

1 Mutual Authentication (M.A.)

2 Alice submits a job via Globus GRAM

3 Delegate proxy certificate

NEESgrid Linux Cluster

NEESgrid Linux Cluster Gatekeeper

RLS

M.A.

M.A.

M.A.

M.A.

GridFTP Server

SRB

Shake table

M.A.: Mutual Authentication
Scalability Issues

- Different sites trust different certificates issuers
  - No way to determine automatically which issuers are trusted
- Headache for users to manage their many certificates, even with help of a credential wallet
- In large projects, an account per user does not scale
  - Resource owners cannot have a local account in place for every potential user
- Authorization may depend on user’s properties
  - E.g., user’s affiliation with a project
  - New authorization certificates: PRIMA, VOMS, CAS, X.509 attribute
Trust Negotiation

- Establish trust iteratively and bilaterally by the disclosure of certificates and by requests for certificates
- Trust is based on parties’ properties
- Every party can define access control policies to control outsiders’ access to their sensitive resources
Step 1: Alice requests a service from Bob

Step 2: Bob discloses his policy for the service

Step 3: Alice discloses her policy for VISA

Step 4: Bob discloses his BBB credential

Step 5: Alice discloses her VISA card credential

Step 6: Bob grants access to the service
PeerTrust Language

Based on guarded distributed logic programs
- Datalog with constraints + lists
- Some new syntax features: @, $, signedBy

“Shake table access manager”:
grant(Job, Operation, Resource) $ Job ←
actingOnBehalf(Job, User) | member(User, Project) @ ’CAS-1’ @ Job,
grant(Project, Operation, Resource) @ ’CAS-2’,
accessManager(Resource, ‘Shake table access manager’),
currentDay(Today),
authorizedDay(Project, Today),
notAlreadyInUse(Resource, Job).
How It Works

To gain access to a grid resource, I must satisfy its access control policy, which describes:
- Which certificates must be presented to gain access to it
- Who is responsible for obtaining and presenting these certificates

To obtain (access to) a certificate, I must satisfy its access control policy, which specifies ... --and so on, recursively--

This leads to the distributed construction of a certified proof tree:
- Root represents access to the grid resource
- Leaves represent disclosed credentials / signed facts
- Internal nodes represent heads of policies / signed rules, children represent literals in their bodies
- Inference rules govern allowable edges in the tree
Grid Negotiation

1. Authentication
2. Request
3. Alice's membership?
4. Alice's membership?
5. Alice BigQuake membership
6. Alice BigQuake membership
7. Access granted
8. Alice’s job: Shakes the table

0. Alice submits a job

Alice Smith

MyProxy Server

Scheduler

Shake Table

Access Manager

Shake table
Current Status

Policy evaluation based on Prolog meta-interpreter
- Implements an SLD resolution-like (goal-driven) negotiation strategy

Embedded in Java application / Java applet
- Imports credentials and policies
- Uses TLS connection for secure communication
Conclusions & Future Work

- Trust negotiation
  - Bilateral, iterative establishment of trust
  - Relies on digital credentials, access control policy for each grid resource

- A proposed approach to provide scalability for the Grid Security Infrastructure

Further work:

- Implement on the Grid and evaluate performance
- Extend to Web Services on the Grid
Questions?