CS 1699
Privacy in the Electronic Society

William Garrison
bill@cs.pitt.edu
6311 Sennott Square
http://cs.pitt.edu/~bill/1699

15: ReBAC and trust management
Today: ReBAC and trust management

We’ll consider **relationship-based** access control (ReBAC)
  • Policies defined over graph of connections between users

**Trust management** is a framework for representing:
  • credentials
  • security policies
  • trust relationships

We’ll look at two forms
  • **SD3** (Secure Dynamically Distributed Datalog)
  • **RT** (Role-based Trust-management)
Relationship-based access control

Social networks can be used as the basis for access control

- Poly-relational: Can encode different types of relationships
  - Patient-physician, parent-child, instructor-student...
- Multi-context: Access decisions depend on context
A social network is a **digraph** with multiple edge types

- $V$ is a finite set of vertices (representing **individuals**)
- $\{R_i\}_{i \in I}$, a family of **binary relations**
  - Each $R_i \subseteq V \times V$ specifies pairs of vertices that participate in relationships of type $i$
- $\mathcal{G}(V, I)$: the set of all social networks over vertices $V$, relations $I$

Let:

- $\mathcal{U}$ the set of users, $\mathcal{R}$ set of resources
- Each access request is over a protected resource $r \in \mathcal{R}$, owned by $u \in \mathcal{U}$, requested by $v \in \mathcal{U}$
- For each resource, $\mathcal{U} \times \mathcal{U} \times \mathcal{G}(\mathcal{U}, I) \rightarrow \mathbb{B}$ defines **access policy**
Syntax for ReBAC access control policies

An access control policy can be any of:

- $\top$, constant true
- $a$, an identifier for the owner themselves
- $\neg \phi$, negation of the policy $\phi$
- $\phi \lor \psi$, disjunction of policies $\phi$ and $\psi$
- $\langle i \rangle \phi$, owner has the $i$ relationship with a vertex, and that vertex is related to the accessor via $\phi$

What do these policies mean?

- $\langle \text{spouse} \rangle a$
- $\langle \neg \text{parent} \rangle a$
- $\neg \langle \text{parent} \rangle \langle \text{parent} \rangle a$
- $\langle \text{parent} \rangle a \lor \langle \text{parent} \rangle \langle \text{siblings} \rangle a \lor \langle \text{parent} \rangle \langle \text{siblings} \rangle \langle \text{spouse} \rangle a$
How might ReBAC be useful for electronic health records?
Trust management, introduction

Trust management is a framework for managing:

- security **policies** (access control)
- security **credentials** (authentication)
- trust relationships (delegation)

Consider an electronic banking system:

- At least k bank officers must approve loans over $1M (**policy**)
- Employees can prove they are bank officers (**credentials**)
- Bank specifies who is allowed to issue officer credentials (**trust**)
BFL Principles

Blaze, Feigenbaum, and Lacy (1996) proposed a set of principles for trust management

- **Unified mechanism**: Policies, credentials, and trust can be expressed in the same language
- **Flexibility**: Expressive enough to support complex relationships
- **Locality of control**: Each entity decides which third party credentials to trust
- **Separation of mechanism from policy**: Mechanism for verifying credentials does not depend on the credentials themselves or the applications that use them
A quick introduction to Datalog

A program is a set of rules; each rule is a **logical implication** (lhs IF rhs)

- \( E(1,2) : - \);
- \( E(2,3) : - \);
- \( T(x,y) : - \ E(x,y); \)
- \( T(x,y) : - \ T(x,z), \ T(z,y); \)

A query asks either a) whether a statement is true; or b) for which values of a variable is a statement true

- \(?T(1,3)\)
  - True
- \( (x) : - \ T(1,x) \)
  - \( x = \{2,3\} \)
SD3 generalizes datalog to refer to relations under others’ control

The syntax $K$\$E$ refers to the relation $E$ under the control of the keyholder of public key $K$

- $T(x,y) : - K$\$E(x,y)$;
- $T(x,y)$ holds if $K$ says $E(x,y)$

This allows, e.g., chains of trust

- $T(x,y) : - K$\$G(z), z$\$E(x,y)$;
  $T(x,y) : - T(x,z), T(z,y)$;
- First line handles key distribution: Say, $K$ vets sub-authorities and publishes $G$, each sub-authority in $K$\$G$ publishes their own _$E$
- Second line is transitivity: Say, if $x$ trusts $z$, and $z$ trusts $y$, $x$ should also trust $y$
SD3 can also represent classical access control

Example Unix policy:

- Group("wheel", "alice") :- ;
  Group("employee", "alice") :- ;
  Group("employee", "bob") :- ;
- Perms("bob", "file /etc/hosts", "modify") :- ;
  Perms(user, "file /etc/motd", "read") :-
    Group("employee", user);
  Perms(user, resource, op) :-
    PKD("alice", k), k$Perms(user, resource, op);

Meaning:

- Alice is in wheel, Alice and Bob are in employee
- Bob can modify /etc/hosts, any employee can read /etc/motd
- Alice can define additional permissions
Which users are employees?
• (user) :- Group(“employee”, user)

What can Bob modify?
• (resource) :- Perms(“bob”, resource, “modify”)

Who can do what with /etc/motd?
• (user, op) :- Perms(user, “file /etc/motd”, op)
RT is a typesafe family of languages for trust management

$RT_0$ provides local roles, hierarchies, delegation, and role intersections

- Attribute assignment
  - Pitt.student $\leftarrow$ Alice
- Delegation of attribute authority
  - Pitt.student $\leftarrow$ PittCS.student
- Attribute inference
  - Epub.access $\leftarrow$ Epub.student
- Attribute-based delegation of authority
  - Epub.student $\leftarrow$ Epub.university.student
- Intersection
  - Epub.access $\leftarrow$ Epub.student $\cap$ ACM.member
RT₁ extends RT₀ with parameterized roles

Main idea: Allow attributes that have fields

Type I:
- A.r(h₁, ..., hₙ) ← D

Type II:
- A.r(h₁, ..., hₙ) ← B.r₁(s₁,...,sₙ)

Type III:
- A.r(h₁, ..., hₙ) ← A.r₁(t₁,...,tₙ).r₂(s₁,...,sₘ)

Type IV:
- A.R ← B₁.R₁ ∩ B₂.R₂ ∩... ∩ Bₖ.Rₖ
$RT_1$ extends $RT_0$ with parameterized roles

Examples:

- Alice is a Pitt (graduate) student with CS major
  - Pitt.student(major="cs", program="graduate") ← Alice
- Epub access is allowed for Pitt graduate students
  - Epub.access ← Pitt.student(program="graduate")
- Epub read access to certain files is allowed for CS students
  - Epub.access(action="read", resource="some_file") ← Epub.university.student(major="cs")
- Instructors can assign grades for students they teach
  - Pitt.canGrade(student=y) ← Pitt.teacherOf(student=y)
Further extensions in the RT framework

$RT_2$: Logical objects
  • Loosely: roles for objects

$RT^T$: Thresholds and separation of duty
  • Threshold: Require agreement among $k$ different principals from a list
  • SoD: Require 2 or more different persons to complete a task
Conclusions

ReBAC is a generalization of RBAC for n-ary relations (rather than only unary roles)

Trust management encapsulates several security issues into a unified framework

- SD3 and RT are based on Datalog
- SD3 is one of the first distributed trust management languages
- RT is very structured, expressive, and typesafe

Next week: Enjoy your break!