The RT Trust Management Framework

Presented by Peter C. Chapin
University of Vermont

December 1, 2004

References

- N. Li, J. Mitchell, and W. Winsborough. *Design of a Role-based Trust-management Framework*. In Proceedings of 2002 IEEE Symposium on Security and Privacy, pages 114–130. IEEE Computer Society Press, May 2002

- M. Blaze, J. Feigenbaum, J. Lacy. *Decentralized Trust Management*. In Proceedings of 1996 IEEE Symposium on Security and Privacy, pages 164–173, May 1996

- D. Ferraiolo and R. Kuhn. *Role-based access controls*. In Proceedings of 15th NIST-NCSC National Computer Security Conference, pages 554–563, Baltimore, MD, October 13-16 1992

Outline

- Trust Management
- Role Based Access Control
- RT: **R**ole Based **T**rust Management

Trust Management

- Traditional access control uses “closed world” assumption: all users are known to the system.
  - Example: Unix or Windows
- In a decentralized, distributed system the closed world assumption is unreasonable.
  - Infeasible to maintain a list of all potential users.
  - Access must be granted based on certified characteristics of users (group membership, age, eye color, etc).

Who's There?

Identity not particularly useful.

Example: We don't know Jill Jones so knowing that she is requesting access isn't helpful. We might rather know that the client

- Is UVM CS faculty **OR**
- Is a UVM CS student **AND** has a GPA $\geq$ 3.5

X.509 and PGP provide identity certificates. Useful in a closed world.

Characteristics of TM Systems I

Trust management systems provide a way of describing. . .

1. Principals: entities to which access can be granted or trust extended. Public keys represent individuals (human and non-human), groups, roles, etc.

2. Actions or permissions: security sensitive operations. Often implicit in a request. Possibly high level. Application specific.

3. Policy: what actions can be done by which principals. Example: UVM students can read, UVM CS students can update.

Characteristics of TM Systems II

1. Credentials: signed certificates. Proves the specified principal possesses a particular characteristic.

2. Trust relationships: who is trusted to certify what. For example, the UVM registrar is trusted to certify UVM student status.

Delegation

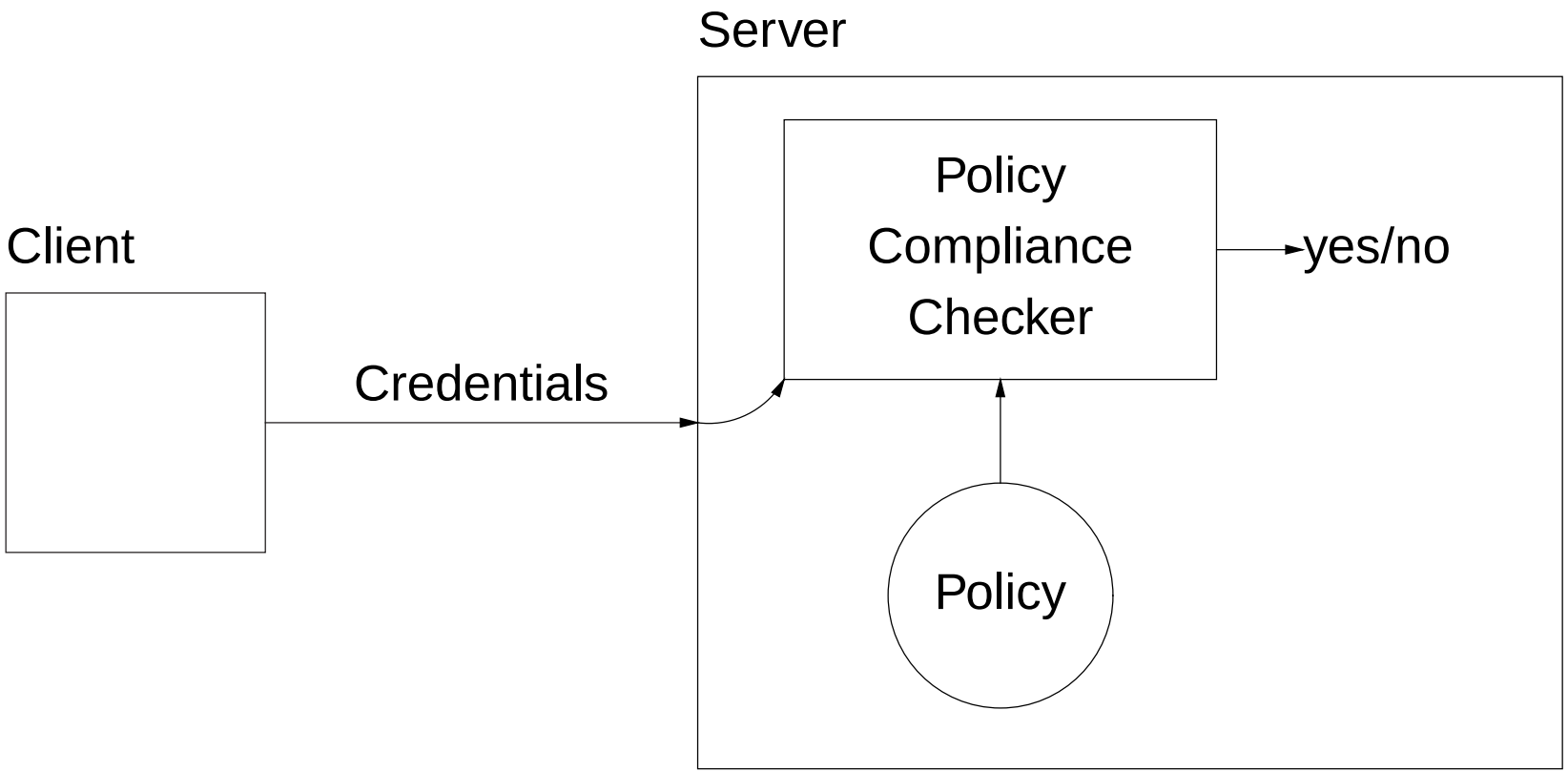
There are two kinds of delegation.

- Delegation of trust: Who is trusted to certify characteristics? Example: Doctors can certify height. The AMA can certify doctors.

- Delegation of authority: Who can act on a principal's behalf in a transaction?

First form is an essential part of trust management. Second form is optional, but often useful.

Anatomy of a Request



Role Based Access Control (RBAC)

Introduced to simplify administration of access control.

- Roles define both users and permissions.
- Easy to look up both role members and role permissions; simply consult the role definition.
- In contrast groups only define users. Permissions are defined separately (on ACLs, for example) and are thus harder to manage.
- RBAC not discretionary. Only creator of role can modify role permissions. Resource owners not necessarily involved.

More RBAC

Roles can be used in several ways depending on the system's security needs.

- Organize users into groups. Example: UVM CS students.
- Represent permissions. Example: Print on network printer. Role members have the indicated permission.
- Represent attributes. Example: People more than 6 feet tall. Role members have the indicated attribute (ABAC).

Some systems allow roles to contain other roles forming role hierarchies.

RT Framework

RT combines ideas from trust management systems and RBAC.

- Trust management systems provide notion of trust delegation.
- RBAC provides notion of flexible roles.
- RT offers several variations: RT_0, RT_1, RT_2, RT^T, RT^D, RT^C, etc. Variations trade off expressiveness and complexity.

RT_0 Roles

- Role names local to principals (like SDSI/SPKI).
- Roles defined by specifying role members.
- Some roles defined as part of local policy, some as signed certificates.
- Role membership is the union of all policy and certificate definitions.

Membership Assertions I

$A.R \leftarrow B$

Principal B is in A's role R. Example: `WS.student` $\leftarrow$ `Alice`

$A.R \leftarrow B.R_1$

Members of B's role R_1 are in A's role R. This is trust delegation. Example: `WS.student` $\leftarrow$ `UVM.student`

Membership Assertions II

$A.R \leftarrow A.R_1.R_2$

A's role R contains all members of B's R_2 role if B is a member of A's R_1 role. Example: `WS.student` $\leftarrow$ `WS.university.student` with an appropriate definition of `WS.university`.

$A.R \leftarrow B_1.R_1 \cap B_2.R_2 \cap \ldots \cap B_n.R_n$

A's role R contains the entities that are members of all the roles $B_1.R_1$, $B_2.R_2$, etc. Example: `WS.student` $\leftarrow$ `UVM.student` $\cap$ `UVM.gradcollege`.

Example

WS's policy:

$$\texttt{WS.readsite} \leftarrow \texttt{WS.student}$$
$$\texttt{WS.student} \leftarrow \texttt{WS.university.student}$$
$$\texttt{WS.university} \leftarrow \texttt{ABU.accredited}$$

Submitted with Alice's request to read the site:

$$\texttt{UVMregistrar.student} \leftarrow \texttt{Alice}$$
$$\texttt{UVM.student} \leftarrow \texttt{UVMregistrar.student}$$
$$\texttt{ABU.accredited} \leftarrow \texttt{UVM}$$

Access is granted.

Datalog

- RT semantics defined in terms of Datalog.
- Datalog is a subset of Prolog that lacks function symbols and negation. Introduced as a database query language supporting recursion.
- RT assertions can be translated into Datalog rules and facts.
- Access is granted if the necessary fact can be proven.

Example (Continued)

```
% Policy
member(ws, readsite, X) :- member(ws, student, X).
member(ws, student, X) :- member(ws, university, Y),
                          member(Y, student, X).
member(ws, university, X) :- member(abu, accredited, X).

% Credentials submitted with request
member(uvmregistrar, student, alice).
member(uvm, student, X) :- member(uvmregistrar, student, X).
member(abu, accredited, uvm).
```

Computing `member(ws, readsite, alice)` returns “yes”.

RT_1

RT_1 adds parameterized roles.

- `WidgetsInc.managerOf(Alice)` $\leftarrow$ `Bob`
- `WidgetsInc.evaluatorOf(?X)` $\leftarrow$ `WidgetsInc.managerOf(?X)`
- `UVM.recentAlumni` $\leftarrow$ `UVM.graduated(?Year:[2001..2004])`
- `UVM.graduated(2001)` $\leftarrow$ `Alice`

Multiple parameters allowed on a role. Parameters are typed and can take on discrete integer, float, or enumeration values.

RT_1^C enriches role parameters by allowing more expressive constraint domains.

Other RTs

- RT_2. Supports parameterized object sets.
- RT^T. Supports threshold and separation of duty policies.
- RT^D. Supports role activations and delegation of those activations.
- RT^C. Supports structured resources (semantics requires Datalog with constraints).

The various options can be mixed: RT_1^D, RT_2^{DT}, etc.

Questions

?