Access Control Models

Lionel Montrieux <lionel@nii.ac.jp>
Definitions and Scoping

What **is** access control, anyway?
The subject is most often assumed to be a human. A **non-person entity (NPE)**, such as an autonomous service or application could also fill the role of the subject. In general, every operation performed by a computer must be done on behalf of some person or organization (in the case of an NPE) with the authority to perform the operation. The term **subject** is used to denote a human or NPE requesting access to an object. [NIST-ABAC]
User != Subject

Don’t expect standards to use these terms consistently
**Authentication** is the act of verifying that the subject has been authorized to use the presented identifier by a trusted identity provider organization. [NIST-ABAC]
Authorization is the decision to permit or deny a subject access to system objects (network, data, application, service, etc.) [NIST-ABAC]
What is Access Control?

- Authentication + Authorisation?
What is Access Control?

- Authentication + Authorisation?
- Authorisation only?
What is Access Control?

- Authentication + Authorisation?
- Authorisation only?
- Identification + authentication + authorisation + access decision?
What is Access Control?

- Authentication + Authorisation? (RBAC)
- Authorisation only? (ABAC - NIST)
- Identification + authentication + authorisation + access decision? (HP, industry)
Access Control Mechanism

The logical component that serves to receive the access request from the subject, to decide, and to enforce the access decision. [NIST-ABAC]
Things I will not talk about

- Policy evaluation
- Delegation
- Federated services
- Authentication
- Performance
- Identity Providers
- PKI
- Trust issues and management
- etc, etc
Notable Models

Security researchers like to come up with access control models a bit too much
Surely, there must be a single, well-understood, clearly formalised model for each major authorisation model, right?
Well, no.
• XACML is full of errors
• RBAC too, but at least it’s easier to formalise [Huynh2014]
Timeline

- 1960’s - 1970’s: multi-user systems
Timeline

- 1960’s - 1970’s: multi-user systems
- 1985: DAC & MAC (TCSEC)
Timeline

- 1960’s - 1970’s: multi-user systems
- 1985: DAC & MAC (TCSEC)
- 1990’s: Roles
Timeline

- 1960’s - 1970’s: multi-user systems
- 1985: DAC & MAC (TCSEC)
- 1990’s: Roles
- 2000-2001: RBAC standard (NIST)
Timeline

- 1960’s - 1970’s: multi-user systems
- 1985: DAC & MAC (TCSEC)
- 1990’s: Roles
- 2000-2001: RBAC standard (NIST)
- 2000’s - now: ABAC
- 2014: First NIST guide for ABAC
DAC

Subject -> Perm.
A **group** is a set of **users**
Role-Based Access Control

[NIST-RBAC]
A role is a job function within the organization that describes the authority and responsibility conferred on a user assigned to the role [Sandhu1996]
A role is a set of permissions
RBAC\textsubscript{1} - Hierarchies

A role inherits permissions
There is More...

Diagram:

- Users
- Roles
- Sessions
- Ops.
- Objects

Connections:

- Users to Roles: UA
- Roles to Sessions: RH
- Sessions to Roles: PA
- Roles to Ops.:
- Ops. to Objects:

Database Tables:

- user_sessions
- session_roles

[Source: NIST-RBAC]
RBAC

UA ⊆ Users x Roles

[Diagram showing the relationship between Users, Roles, Permissions, Operations, Objects, Sessions, and the relationships user_sessions, session_roles, RH, UA, and PA]
$PA \subseteq Permissions \times Roles$

[NIST-RBAC]
assigned permissions(r:Roles) → 2Permissions
\[
\text{assigned permissions}(r) = \{ p \in \text{Permissions} \mid (p, r) \in PA \}
\]
$Ob(p: Permissions) \rightarrow \{op \subseteq \text{Ops}\}$
$$Ob(p: Permissions) \rightarrow \{ob \subseteq Objects\}$$
user sessions (u: Users) → 2Sessions
session roles \((s: \text{Sessions}) \rightarrow 2\text{Roles}\)

\[
\text{session roles } (s_i) \subseteq \{r \in \text{Roles} \mid (\text{session users } (s_i), r) \in \text{UA}\}
\]
avail_session_perms(s:Sessions) → 2Permissions

[NIST-RBAC]
RH ⊆ Roles × Roles is a partial order on Roles called the inheritance relation, written as ≥, where r₁ ≥ r₂ only if all permissions of r₂ are also permissions of r₁, and all users of r₁ are also users of r₂. Formally: r₁ ≥ r₂ ⇒

\[
\text{authorized_permissions}(r₂) \subseteq \text{authorized_permissions}(r₁) \land \\
\text{authorized_users}(r₁) \subseteq \text{authorized_users}(r₂).
\]
RBAC

\[ \text{authorized\_users}(r: \text{Roles}) \rightarrow 2\text{Users} \]
\[ \text{authorized\_users}(r)=\{u \in \text{Users}|r' \geq r (u,r') \in UA\} \]

\[ \text{authorized\_permissions}(r: \text{Roles}) \rightarrow 2\text{Permissions} \]
\[ \text{authorized\_permissions}(r)=\{p \in \text{Permissions}|r' \geq r,(p,r') \in PA\} \]

[NIST-RBAC]
ABAC

Attribute-Based Access Control
A logical access control methodology where authorisation to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes. [NIST-ABAC]
An attribute is a function which takes an entity such as a user and returns a specific value from its range. An attribute range is determined by its scope and type. The scope of an attribute is given by a finite set of atomic values. [Jin2014]
$ABAC_a$

[Image of a diagram showing the relationships between U, S, O, UA, SA, OA, and Authz.

References: [Jin2012, Jin2014]
ABAC_\alpha_
**Constraints** are functions which return true when conditions are satisfied and false otherwise. Security architects configure constraints via policy languages. [Jin2014]
Authorization policies are two-valued boolean functions which are evaluated for each access decision. An authorization policy for a specific permission takes a subject, an object and returns true or false based on attribute values. More generally, access decision may be three-valued, possibly returning “don’t know” in addition to true and false. This is appropriate in multi-policy systems. [Jin2014]
ABAC_α - What can it Express?

- DAC
- MAC
- RBAC_0
- RBAC_1

[Jin2012, Jin2014]
[Jin2014]
• Context attributes

[Jin2014]
ABAC$\beta$ - Extensions

- Context attributes
- Subject attribute constraint policy at creation time is different from modification time

[Jin2014]
ABAC$_{\beta}$ - Extensions

- Context attributes
- Subject attribute constraint policy at creation time is different from modification time
- Subject attribute constrained by attributes of subjects created by the same user

[Jin2014]
ABAC$\beta$ - Extensions

- Context attributes
- Subject attribute constraint policy at creation time is different from modification time
- Subject attribute constrained by attributes of subjects created by the same user
- Enhanced policy specification language

[Jin2014]
ABAC_β - Extensions

- Context attributes
- Subject attribute constraint policy at creation time is different from modification time
- Subject attribute constrained by attributes of subjects created by the same user
- Enhanced policy specification language
- Meta-attributes

[Jin2014]
Policy Languages
XACML Policy

- PolicySet
  - Policy Combination Alg.
  - Obligation
  - Advice
  - PolicySet
  - Policy
    - Obligation
    - Advice
    - Rule Combination Alg.
    - Target
    - Rule
      - Obligation
      - Advice
      - Effect
      - Target
        - Subject
        - Resource
        - Action
        - Environment
        - Condition
XACML Request

- Request
  - Subject attributes
  - Resource attributes
  - Action attributes
  - Environment attributes
Processing Requests

How does it work?

WE WANT... A SHRUBBERY!!
Access Control in Practice

Turns out, many things can go wrong
Practice Time!

Let’s write a policy and a few requests
Thank you!

Any questions?
References

• Doodles by J. Hardaway (inspired by Monty Python’s Holy Grail), with permission
• Dictionary http://tinyurl.com/qep8w94 (CC by)
• Keyboard http://tinyurl.com/oyqob8c (CC by-nc)
• Padlock http://tinyurl.com/pmp2yjc (CC by-sa)
• Turnstile http://tinyurl.com/qjnl296 (CC by-nc-nd)