Lecture 7

Monday, 26 September 2022 21:14

RBAC requires engineering of roles

- role identification
- permission identification
  - least privilege
  - separation of duty

RBAC was designed for simple systems

role designing + engineering is difficult & expensive

changes in business can lead to security inconsistencies

RBAC requires attention all the time

RBAC is not context-aware or dynamic

i.e. no restrictions involving location, time

RBAC does not really scale...

- large number of roles required
- RBAC cannot deal with relations

---

attribute-based access control

assign many attributes to users/resources

instead of many users to role

ABAC determines authorization by evaluating attributes against policy / rules

attributes, define specific aspects of the subject / object / environment conditions / requested actions, etc...

environmental conditions are dynamic and independent of subject / object

ABAC allows both positive and negative authorizations (i.e. negation)

form: if condition AND/OR condition then Permit/Deny

ABAC is more granular and easier to setup than RBAC

however, ABAC is more difficult to administer

ABAC is context-aware but can be more reliable than RBAC

questions:

- who defines policies?
- do policies reflect access requirements?
- from where are attributes retrieved?

problem: control over data is often decentralized

and this control may be delegated over the policy definition

how to handle multiple policies over one object?

e.g. with priority (access permit overrides other rules)

how to verify whether a policy meets its requirements?

in ABAC: attributes can be retrieved from external sources

what if these sources are unavailable? or an incorrect value is provided?