Authorization - Policy

Bill Cheng

http://merlot.usc.edu/cs530-s10

Authorization

Final goal of system security
- determine whether to allow an operation
  - authentication
  - audit - so that you can change policy to keep the bad guys out

Depends upon
- policy - rules followed by the system
- possibly authentication
- policy can be based on identity
- other characteristics - e.g., time of day, network threat condition, system load

The Role of Policy in Security Architecture

*Policy* - defines what is allowed and how the system and security mechanisms should act
(misconfiguration - policy does not reflect intent)

**Enforced By**

**Mechanism** - provides protection
interprets/evaluates policy
(firewalls, ID, access control, confidentiality, integrity)

**Implemented As**

**Software** - which must be implemented correctly and according to sound software engineering principles

Policy: Review - The Access Matrix

Policy represented by an Access Matrix
- also called Access Control Matrix
- one row per object
- one column per subject/principle
- tabulates permissions
- but implemented by:
  - capability list (like a key ring)
  - Access Control List (ACL)
    - recall that it’s harder to determine who has access with ACL

Policy models: Bell-LaPadula

- *Discretionary* policy
  - based on Access Matrix - owner of an object can determine who has access

- *Mandatory* policy
  - owner of an object does not get to decide who has access
  - Top Secret, Secret, Confidential, Unclassified
  - “property: S can write O if and only if Level S \(\leq\) Level O
    - write UP, read DOWN
  - it’s possible that I can create a file that I cannot read
  - create categories so that some members in a class cannot see some documents
  - this approach tries to minimize the speed of secret leaks

(more models in Bishop’s book, e.g., integrity policy)

Role Based Access Control

- In a way, similar to groups in UNIX, but more general
  - in UNIX, an object can belong to only a single group, inconvenient to create dynamic groups

Three phases
- administration
- session management
- access checking

Typical policies
- object policies fairly static
- user’s roles can change
  - but no need to list all objects to which users has access

Maps to typical organizational policies
- can implement separation of roles
Security is More Than Mix of Point Solutions

Today’s security tools work with no coordinated policy
- firewalls and Virtual Private Networks
- authentication and Public Key Infrastructure
- intrusion detection and limited response

We need better coordination
- intrusion response affected at firewalls, VPN’s and applications
- not just who can access what, but policy says what kind
  of encryption to use, when to notify ID systems

Tools should implement coordinated policies
- policies originate from multiple sources
- policies should adapt to dynamic threat conditions
- policies should adapt to dynamic policy changes triggered
  by activities like September 11th response

Policies Originate from Multiple Sources

Discretionary policies associated with objects
- e.g., one module for reading .ssh files and one module
  for reading .htaccess files

Local system policies merged with object policies
- broadening or narrowing allowed access - can ignore
discretionary policy
- e.g., deny all web accesses from certain domains

Policies imported from policy/state issuers
- example of policy issuers is virus checker from Network
  Associates or Symantec
- example of state issuers is HIPAA - healthcare related
  policy for healthcare providers

Policies Originate from Multiple Sources (Cont...)

Policies imported from policy/state issuers (cont...)
- ID system issues state credentials
  - these credentials may embed policy as well

Policies embedded in credentials
- these policies attach to user/process credentials and
  apply to access by only specific processes
  - e.g., extra audit required from outsiders
  - this also allows chaining

Policies evaluated remotely
- credential issuers (e.g., authentication and authorization
  servers) evaluate policies to decide which
  credentials to issue.

Policies Origins Summary

HIPAA, other legislation
- e.g., access to student records
Privacy statements
- need to know how it is actually enforced
Discretionary policies
Mandatory policies (e.g. classification)
Business policies

GAA-API: Integration through Authorization

GAA: Generic Authorization and Access-control
Focus integration efforts on authorization and the
management of policies used in the authorization decision
- not really new - this is a reference monitor (as in TOPS-20
  and MULTICS)
- applications shouldn’t care about authentication or identity
  - separate policy from mechanism
- authorization may be easier to integrate with applications
- hide the calls to individual security services
  - e.g., key management, authentication, encryption, audit
  can perform adaptive audit
  - dynamic policy
  - when ID detects something, start collecting additional
    information or start requiring authentication
    even for internal users

GAA-API

Sometimes it is not possible to plug in security at low level
- need information at the application level
  - Ex: SSL is in the lower layer, it cannot deal with user
    certificates

GAA-API: application just asks if something is allowed
- return value is either yes, no, or maybe
  - maybe means you need additional things, e.g., network
    source address must come from a certain domain (this
    information, again, may not be available at lower layers)

Subject/principle is represented by a Security Context (SC)
why not an identity?
- because sometimes it’s not necessary, e.g., to access
  this, pay $5 (no authentication)
GAA-API (Cont...)  
- EACL (extended ACL)  
  - the language used by GAA  
  - extended to include information such as:  
    - time of day  
    - network threat condition  
    - system load

Authorization and Integrated Security Services  
Integration of dynamic security services creates feedback path enabling effective response to attacks

Generic Authorization and Access-control API (GAA-API)  
- Allows applications to use the security infrastructure to implement security policies  
- `gaa_get_object_policy_info()` function called before other GAA-API routines which require a handle to object EACL to identify EACLs on which to operate  
- can interpret existing policy databases  
- `gaa_check_authorization()` function tells application whether requested operation is authorized, or if additional application specific checks are required

Three Phases of Condition Evaluation  
- `gaa_post_execution_actions()`  
- `gaa_execution_control()`  
- `gaa_check_authorization()`  
- `gaa_get_object_policy_info()`  

Communicating threat conditions  
- Threat conditions and new policies carried in signed certificates  
  - added info in authentication credentials  
  - threat condition credential signed by ID system  
  - it is often done to run System High - always assumes that threat condition is RED, only change if received signed certificate to say that it’s no longer RED  
- Base conditions require presentation or availability of credential  
  - matching the condition brings in additional policy elements

Integrating Security Services  
- The API calls must be made by applications  
  - this is a major undertaking, but one which must be done no matter how one chooses to do authorization.  
  - These calls are at the control points in the applications  
    - they occur at auditable events, and this is where records should be generated for ID systems  
    - they occur at the places where one needs to consider dynamic network threat conditions  
    - adaptive policies use such information from ID systems  
    - they occur at the right point for billable events
Advances Needed in Policy

- Ability to merge & apply policies from many sources
  - legislated policies
  - organizational policies
  - agreed upon constraints
- Integration of policy evaluation with applications
  - so that policies can be uniformly enforced
- Support for adaptive policies is critical
  - allows response to attack or suspicion
- Policies must manage use of security services
  - what to encrypt, when to sign, what to audit
  - hide these details from the application developer

What Dynamic Policies Enable

- Dynamic policy evaluation enables response to attacks:
  - lockdown system (or bump up security) if attack is detected
  - establish quarantines by changing policy to establish isolated virtual networks dynamically
  - allow increased access between coalition members as new coalitions are formed or membership changes to respond to unexpected events
    - e.g., homeland security
    - e.g., open things up - sharing is allowed only when certain credentials have been received

Demo Scenario - LockDown

You have an isolated local area network with mixed access to web services (some clients authenticated, some not)

You need to allow incoming authenticated SSH or IPSec connections

When such connections are active, you want to lock down your servers and require stronger authentication and confidentiality protection on all accesses within the network
Demo Scenario - LockDown (Cont...)
- But how do you know if someone is connecting from the outside?
- You need integrated solutions
- The scenario is like having a visitor in a classified area
  - Security can be inconvenient

But how do you know if someone is connecting from the outside?
The scenario is like having a visitor in a classified area
- Security can be inconvenient

Proxies
- A proxy allows a second principal to operate with the rights and privileges of the principal that issued the proxy
- Existing authentication credentials
- Too much privilege and too easily propagated
- Restricted proxies
  - By placing conditions on the use of proxies, they form the basis of a flexible authorization mechanism

Proxies Example
- Ex: I want to print to this printer
  - Printer only accepts authorization from authorization server
  - Talk to authorization server
  - Authorization server says "maybe" with condition in credential
  - Since you are a visitor, you must pay
  - Authorization server generates proxy, includes policy, returns to user as capability

Restricted Proxies
- Two kinds of proxies
  - Proxy key needed to exercise bearer proxy
  - A bearer proxy can be used by anyone
  - Restrictions limit use of a delegate proxy
- Restrictions limit authorized operations
  - Individual objects
  - Additional conditions
  - When, where, how
  - Additional audit records may be produced

Mechanisms Summary
- Access Matrix
  - Access Control List (ACL)
  - Capability list (key ring)
- Web server
  - .htaccess
- Unix file system
  - Basically ACL
  - At login, look up which groups you belong, associate that list with your login process (this is like capability)
- When you open a file, the file descriptor is like capability?
- SSH authorized key files
- Restricted proxies, extended certificates
- Group membership
- Payment

Summary
- Policies naturally originate in multiple places
- Future systems need to deal with this
- Deployment of secure systems requires coordination of policy across countermeasures
- Effective response requires support for dynamic policy evaluation
- Such policies can coordinate the collection of data used as input for subsequent attack analysis