Traditional Access Control

Policy
login, ID=Tom

Authorization Mechanism
grant/deny

Request
bom.isi.edu, login, Tom

bom.isi.edu
Motivation

- Support of the advanced policies that allow actions when security violations are suspected or detected.

- Support policy enforcement at various time stages of the requested action.

- Simplify integration of related security services, such as authentication, intrusion detection, audit and notification with applications.

- Facilitate authorization decisions for applications.

- Provide generic policy evaluation environment.

- Provide a uniform integration model.

- Aim for extensibility to avoid the need to redesign the system in the future.
“Tom can run a process on host bom.isi.edu. If the request fails, a notification must be sent to a system administrator. The process must not consume more than 20% of the CPU. An audit record about the completed process must be generated.”
**Conditions**

- **Access identity** specifies authenticated user identity.
- **Authentication method** specifies mechanisms suitable for authentication.
- **Payment** specifies currency and amount.
- **Time** time periods expressed as time of day or days of week when access is granted.
- **Location** access is granted to principals connecting from specific hosts.
- **Notification** enables automatic generation of notification messages.
- **Audit** enables automatic generation of application level audit data.
- **System Threat Level** specifies system threat level, e.g., high, medium or low.
- **Threshold** specifies allowable threshold.
System Variables

- **System Variable** represents a data item of particular type, e.g., a file, a message or a record in a database.

- Assume that there exists a set of software components that accesses system variables of particular type.

- Assume that each software component has abstract operations:
  - \textit{Read}(X) returns the value of the system variable \(X\).
  - \textit{Write}(X, v\_new) assigns a new value \(v\_new\) to the system variable \(X\).

- \textit{Read}(X) and \textit{Write}(X, v\_new) operations return T (success), F (failure) or U (uncertain).
**Taxonomy of Conditions**

- **pre-conditions**: time, access identity
- **request-result-conditions**: audit
- **mid-conditions**: threshold
- **post-conditions**: notification

### Conditions

#### Read Conditions
- \( X \text{ op } P \)
- \( \text{op} \) can represent:
  - numerical comparison
  - string matching
  - regular expression matching
  - set-theoretic comparison
  - delegation

#### Write Conditions
- \( X, \text{ new_value} \)
- on success/failure

#### Opague Conditions
- ?
“Tom can run a process on host bom.isi.edu. If the request fails, a notification must be sent to a system administrator. The process must not consume more than 20% of the CPU. An audit record about the completed process must be generated.”
Extended Access Control List (EACL)

- **Condition block** defines a conjunction of a set of conditions. Conditions are evaluated consecutively.

- **EACL entry** consists of a positive or negative access right and four condition blocks.

- **EACL** consists of an ordered set of disjunctive EACL entries.

- Composition of policy statements from different security domains.
Three-phase Policy Enforcement

EACL
- shut_down, ID=Joe
- login, ID=Tom, audit
- login, day {Mon,Tue}

Request
- bom.isi.edu, login, Tom

GAA-API
- gaa_get_object_policy_info()
- gaa_check_authorization() T/F/U
- gaa_execution_control() T/F/U
- gaa_post_execution_actions() T/F/U

System State
- Read()
- Write()
GAA-API Features

- Extensibility.
  The GAA-API allows addition of modules to support application-defined condition evaluation, policy retrieval and credential management interfaces.

- Support for specific conditions.
  If generic conditions supported by GAA-API are not sufficient, applications specify their own conditions.

- Non-EACL Policy evaluation:
  1. Translate native policy to EACL format using specific conditions. Advantages: negative/positive authorizations; generic conditions; three-phase enforcement model; and policy composition framework.

  2. Whole policy as one condition. Advantages: composition framework (additional policies can be specified using EACLS).

- GAA-API combines policy enforcement and intrusion/misuse detection and response.
GAA-API Current Status

- The GAA-API is implemented as a C-library.
- Code is available at: http://www.isi.edu/gost/info/gaaapi/source
- We integrated the API with Keynote system to support delegation.
- Integration with Grid Security Infrastructure (GSI).
- GAA-API is integrated with Apache server and ssh.
- Integration with other applications including FreeSwan IPsec, Snort.
Current Work: DEFCN Project

- **Objectives**
  - Definition of policies that adapt to system threat conditions.
  - Enable intrusion detection, notification and auditing mechanisms to adapt recorded detail based on threat conditions.
  - Simplify integration of fine grained access control and related security services with applications.

- **Benefits**
  - Real-time response to state generated by Intrusion Detection System (IDS).
  - Application-level intrusion and misuse detection capabilities.
  - Fine-tuning of IDS.
    - adapt level of detail
    - find more attacks
    - reduce # of false positives
Network Lockdown

# EACL entry 1
pos_access_right apache read
pre_cond_access_sys_thr_level generic =low

# EACL entry 2
pos_access_right apache read
pre_cond_access_id_USER apache *

GAA-API

a.html, read

NIDS

Tatyana Ryutov
Copyright 2002, USC/ISI.
# EACL entry 1
pos_access_right  apache  read
pre_cond_access_sys_thr_level  generic  =low

# EACL entry 2
pos_access_right  apache  read
pre_cond_access_id_USER  apache  *

NIDS
a.html

GAA-API

a.html, read
Application-level Intrusion Detection

# EACL entry 1
neg_access_right apache *
pre_cond_GROUP local BadGuys

# EACL entry 2
neg_access_right apache *
pre_cond_regex gnu "phf*, test-cgi"
rr_cond_notify local on:failure,sysadmin
rr_con_update_log local on:failure/BadGuys/IP

# EACL entry 3
pos_access_right apache *

cgi-bin/phf.cgi, execute
NIDS fine-tuning

# EACL entry 1
pos_access_right ssh host_login
test_cond_location IP 10.1.1.0-10.1.200.255
rr_cond_notify generic on:failure/NIDS/info:IP

bom.isi.edu

bom.isi.edu, host_login, Tom

GAA-API

IP = 10.2.1.0
Related Work

• **PolicyMaker** Supports restricted set of policies. Neither supports negative authorizations nor adaptive policies. Closed system.

• **OASIS**
  RBAC system not suited to representing complex policies and those that include mandatory access control.

• **Ponder** supports obligation policies; separate policies and enforcement mechanisms for different types of policies.

• **Minsky & Ungureanu** policy enforcement is based on regulated message exchange.
Future Directions

• Further investigation of the execution control phase.

• Support of concurrent requests, replicated policy evaluation mechanisms and concurrent evaluation of conditions in one policy statement.

• Decentralized policy enforcement.

• Application integration: we are looking to integrate with NTFS.
Conclusions

- The policies should integrate auditing, intrusion detection, authentication and notification services and allow more efficient utilization of these services.

- The recognition that distributed services should be more concerned with the authorization of the requested access rather than with the identity of the entity of the requester.

- The recognition that the points of the policy enforcement may include three time phases.

- GAA-API that combines fine-grained access control and application-level intrusion/misuse detection and response.