Information security is fundamentally about managing
- authorization and
- trust
so as to manage risk
THE OM-AM WAY

<table>
<thead>
<tr>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objectives</strong></td>
<td><strong>Assurance</strong></td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Mechanism</strong></td>
<td></td>
</tr>
</tbody>
</table>

LAYERS AND LAYERS

- Multics rings
- Layered abstractions
- Waterfall model
- Network protocol stacks
- Napolean layers
- RoFi layers
- OM-AM
- etcetera
OM-AM AND MANDATORY ACCESS CONTROL (MAC)

What?
- No information leakage
- Lattices (Bell-LaPadula)
- Security kernel
- Security labels

How?

OM-AM AND DISCRETIONARY ACCESS CONTROL (DAC)

What?
- Owner-based discretion
- Numerous
- Numerous
- ACLs, Capabilities, etc

How?
OM-AM AND ROLE-BASED ACCESS CONTROL (RBAC)

What?

Objective neutral

RBAC96, ARBAC97, etc.

user-pull, server-pull, etc.

certificates, tickets, PACs, etc.

How?

DISTRIBUTED RBAC (DRBAC) CASE STUDY

- Approximately a dozen physical sites
- Approximately 2-3 simulation models/site
- Fewer than 100 roles structured in a very shallow hierarchy
  - A subset of roles is used in any single simulation model
- Fewer than 100 users
- A user uses only one role at a time
  - Convenient but not critical
- Moderate rate of change
DISTRIBUTED RBAC (DRBAC) CASE STUDY

- **Permission-role assignment**
  - Locally determined at each simulation model

- **User-role assignment**
  - A user can be assigned to a role if and only if all simulation models using that role agree
  - A user is revoked from a role if and only if any simulation model using that role revokes the user

---

DISTRIBUTED RBAC (DRBAC) CASE STUDY

- Each simulation model has a security administrator role authorized to carry out these administrative tasks
- A simulation model can assign permissions to a role X at any time
  - even if X is previously unused in that simulation model
- Consequently any simulation model can revoke any user from any role!
Each session has a single role
- \( SM = \{sm1, \ldots, smk\} \), simulation models
- \( OP = \{op1, \ldots, opl\} \), operations
- \( P = SM \times OP \), permissions
- \( SMA = \{sma1, \ldots, smk\} \), administrative roles
- \( R \cap SMA = \emptyset \)
- Admin: \( SM \leftrightarrow SMA \)
MODEL CUSTOMIZATION

- Can formalize the administrative rules given earlier
- For each simulation model designate a unique user to be the chief security administrator who is authorized to assign and revoke users from the security administrator role for that model

DRBAC ARCHITECTURES

- Permission-role
  - Enforced locally at each simulation model
- Permission-role administration
  - Enforced locally at each simulation model
  - May need to communicate to other simulation models
- User-role
  - See following slides
- User-role administration
  - Centralized or decentralized
SERVER MIRROR

Client  Server

User-role
Authorization
Server

SERVER-PULL

Client  Server

User-role
Authorization
Server
USER-PULL

Client → Server

User-role Authorization Server

PROXY-BASED

Client → Proxy Server → Server

User-role Authorization Server
Secure Attribute Services on the Web

- **WWW (World Wide Web)**
  - widely used for electronic commerce and business
  - supports synthesis of technologies
  - mostly, Web servers use identity-based access control
    - scalability problem
Background

- **An attribute**
  - a particular property of an entity
    - e.g., role, identity, SSN, clearance, etc.

- **If attributes are provided securely,**
  - Web servers can use those attributes
    - e.g., authentication, authorization, access control, electronic commerce, etc.

- **A successful marriage of the Web and secure attribute services is required**

User-Pull Architecture

*Authentication information can be either user-based or host-based.*
User-Pull Architecture

- Each user
  - pulls appropriate attributes from the Attribute Server
  - presents attributes and authentication information to Web servers
- Each Web server
  - requires both identification and attributes from users
- High performance
  - No new connections for attributes

Server-Pull Architecture

*Authentication Information can be either user-based or host-based.*
Related Technologies

* Cookies
  - in widespread current use for maintaining state of HTTP
  - becoming standard
  - not secure
* Public-Key Certificates (X.509)
  - support security on the Web based on PKI
  - standard
  - simply, bind users to keys
  - have the ability to be extended

Cookies

<table>
<thead>
<tr>
<th>Domain</th>
<th>Flag</th>
<th>Path</th>
<th>Cookie_Name</th>
<th>Cookie_Value</th>
<th>Secure</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Name</td>
<td>Alice</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Role</td>
<td>manager</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
</tbody>
</table>
Security Threats to Cookies

- Cookies are not secure
  - No authentication
  - No integrity
  - No confidentiality
- can be easily attacked by
  - Network Security Threats
  - End-System Threats
  - Cookie Harvesting Threats

Secure Cookies on the Web

*Sensitive fields can be encrypted in the cookies.
** Seal of Cookies can be either MAC or signed message digest of cookies.
Note: Pwd_Cookie can be replaced with one of the other authentication cookies in Figure 4.

© Ravi Sandhu 2001
A Set of Secure Cookies

How to Use Secure Cookies

Pswd_Cookie can be replaced with one of the other authentication cookies in Figure 4.1
## Applications of Secure Cookies

- User Authentication
- Electronic Transaction
- Eliminating Single-Point Failure
- Pay-per-Access
- Attribute-based Access Control

## Authentication Cookies

<table>
<thead>
<tr>
<th>Cookie</th>
<th>Domain</th>
<th>Flag</th>
<th>Path</th>
<th>Cookie_Name</th>
<th>Cookie_Value</th>
<th>Secure</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>IP_Cookie</td>
<td>129.174.100.88</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Pwd_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Pwd_Cookie</td>
<td>hashed_password</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>KT_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Kerberos_Ticket</td>
<td>(Alice, K_{ca})Ks</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Sign_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Sign_Cookie</td>
<td>Signature_of_Alice</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
</tbody>
</table>
Server-Pull Architecture

- Each user
  - presents only authentication information to Web servers
- Each Web server
  - pulls users’ attributes from the Attribute Server
- Authentication information and attribute do not go together
- More convenient for users
- Less convenient for Web servers

Secure Cookies for Electronic Transactions

*Sensitive fields can be encrypted in the cookies.
**Salt of Cookie can be either MAC or signed message digest of cookies.
Note: Pawd_Cookie can be replaced with one of the other authentication cookies in Figure 4.1
Kerberos-Based Authentication by Secure Cookies

1. Request TGT
2. TGT_Cookie = [TGT, S_A]K_{KDC}
3. TGT_Cookie = TGT
4. KT_Cookie = Tc_s
   KC_Cookie = (K_{CS}, Bob|S_A)

TGT = [S_A, Alice]K_{KDC}
TSK_Cookie = [timestamp|S_A, Alice, Bob
Tc_s = [Alice, K_{CS}]K_s (ticket to Bob)
TSS_Cookie = [timestamp]K_{CS}
TSS'_Cookie = [timestamp+1]K_{CS}

Server (Bob)

Client (Alice)

KDC

Secure Cookies for Pay-Per-Access

- Name_Cookie
  - Domain:HOME; Name:Name_Cookie; Value:Alice*
  - Security:FALSE
  - Expires:12/31/99

- Ticket_Cookie
  - Domain:HOME; Name:Ticket_Cookie
  - Value:ID:56&Expires:10*
  - valid_date:06/09/99
  - Security:FALSE
  - Expires:12/31/99

- Life_Cookie
  - Domain:HOME; Name:Life_Cookie
  - Expires:12/31/99
  - Security:FALSE

- Pwd_Cookie
  - Domain:HOME; Name:Pwd_Cookie
  - Value:hashed_password
  - Security:FALSE
  - Expires:12/31/99

- Key_Cookie
  - Domain:HOME; Name:Key_Cookie
  - Value:encrypted_key*
  - Security:FALSE
  - Expires:12/31/99

- Seal_Cookie
  - Domain:HOME; Name:Seal_Cookie
  - Value:encrypted_data
  - Security:TRUE
  - Expires:12/31/99

* Sensitive fields can be encrypted in the cookies.
** Seal of Cookies can be either MAC or signed message digest of cookies.
Note: Pwd_Cookie can be replaced with one of the other authentication cookies in Figure 4.1
Secure Cookies for RBAC

<table>
<thead>
<tr>
<th>Name</th>
<th>Flag</th>
<th>Path</th>
<th>Cookie Name</th>
<th>Cookie Value</th>
<th>Secure</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>TRUE</td>
<td>/</td>
<td>Name</td>
<td>Alice</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Role</td>
<td>TRUE</td>
<td>/</td>
<td>Role</td>
<td>Manager</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Life</td>
<td>TRUE</td>
<td>/</td>
<td>Life_Cookie</td>
<td>12/31/99</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Pwd</td>
<td>TRUE</td>
<td>/</td>
<td>Pwd_Cookie</td>
<td>Encrypted_Password</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>IP</td>
<td>TRUE</td>
<td>/</td>
<td>IP_Cookie</td>
<td>129.174.142.88</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Seal</td>
<td>TRUE</td>
<td>/</td>
<td>Seal_Cookie</td>
<td></td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
</tbody>
</table>

Cookie_Issuer Signs on the Cookies

* Hash of the passwords is an alternative as the content of the Pwd_Cookie.

RBAC on the Web by Secure Cookies

Assigning Roles & Creating Secure Cookies

Verifying Secure Cookies & RBAC

© Ravi Sandhu 2001
X.509 Certificate

- Digitally signed by a certificate authority
  - to confirm the information in the certificate belongs to the holder of the corresponding private key

- Contents
  - version, serial number, subject, validity period, issuer, optional fields (v2)
  - subject’s public key and algorithm info.
  - extension fields (v3)
  - digital signature of CA

- Binding users to keys
- Certificate Revocation List (CRL)

Certificate Content:

certificate:
  version: 3 (0x05)
  serialNumber: 5 (0x05)
  signatureAlgorithm: MD5 With RSA Encryption
  issuer: CN=CSJ, list.gmu.edu, O=ISL, C=US
  validity:
    notBefore: 2001-10-25 00:00:00
    notAfter: Never
  subject: CN=CSJ, list.gmu.edu, O=ISL, C=US
  subjectPublicKeyInfo:
    publicExponent: 0x10001
    modulus: 0x79e3c58c4f28a1e7a09ee8181e192f86a8b10f8b4d3ad3f03
    publicModulus: 0x79e3c58c4f28a1e7a09ee8181e192f86a8b10f8b4d3ad3f03

  signature:
    algorithm: MD5 With RSA Encryption
    signature: 0x79e3c58c4f28a1e7a09ee8181e192f86a8b10f8b4d3ad3f03

© Ravi Sandhu 2001
Smart Certificates

❖ Short-Lived Lifetime
  ➢ More secure
    • typical validity period for X.509 is months (years)
    • users may leave copies of the corresponding keys behind
    • the longer-lived certificates have a higher probability of being attacked
  ➢ No Certificate Revocation List (CRL)
    • simple and less expensive PKI

Smart Certificates

❖ Containing Attributes Securely
  ➢ Web servers can use secure attributes for their purposes
  ➢ Each authority has independent control on the corresponding information
    • basic certificate (containing identity information)
    • each attribute can be added, changed, revoked, or re-issued by the appropriate authority
      – e.g., role, credit card number, clearance, etc.
  ➢ Short-lived certificate can remove CRLs
Separate CAs in a Certificate

Smart Certificate

- Postdated Certificates
  - The certificate becomes valid at some time in the future
  - possible to make a smart certificate valid for a set of duration
  - supports convenience

- Confidentiality
  - Sensitive information can be
    - encrypted in smart certificates
      - e.g. passwords, credit card numbers, etc.
Applications of Smart Certificates

- On-Duty Control
- Compatible with X.509
- User Authentication
- Electronic Transaction
- Eliminating Single-Point Failure
- Pay-per-Access
- Attribute-based Access Control
Injecting RBAC to Secure a Web-based Workflow System

Gail-Joon Ahn and Ravi Sandhu
George Mason University

Myong Kang and Joon Park
Naval Research Laboratory

WORKFLOW MANAGEMENT SYSTEMS

- Control and coordinate processes that may be processed by different processing entities
- Received much attention
- Marriage with Web technology
- Minimal security services
OBJECTIVE

- Inject role-based access control (RBAC) into an existing web-based workflow system

WHY RBAC?

- A mechanism which allows and promotes an organization-specific access control policy based on roles
- Has become widely accepted as the proven technology
SIMPLIFIED RBAC MODEL

ROLE-BASED SECURE WORKFLOW SYSTEM

- Workflow Design Tool
- Workflow (WF) System
- Role Server
BASIC COMPONENTS

Role Server

- user-role assignment
- role-hierarchy
- user-role DB

Certificate server

NRL design tool

CORBA IIOP

WF system

client

ARCHITECTURES

- USER-PULL STYLE
- SERVER-PULL STYLE
**NRL (Naval Research Lab.)**

**DESIGN TOOL**

- design workflow model
- create role and role hierarchies
- assign role to task
- exporting role hierarchies to role server

---

**NRL DESIGN TOOL (Cont’d)**

Platform: Windows NT, JDK 1.2

© Ravi Sandhu 2001
WORKFLOW SYSTEM

- each task server is web server
- user should present client authentication certificate
- user’s privilege is authorized by content of certificate (specially client’s role information)

ROLE AUTHORIZATION ON WORKFLOW SYSTEM

1. access the resource
2. get client certificate
2.1 retrieve role information
2.2 check authorization status
3. read resource
4. display resource

resources

Task Server (Web Server)

client
ROLE SERVER

- User Role Assignment
- Certificate Server

USER ROLE ASSIGNMENT

- maintain role hierarchies and user database
- assign users to roles
- generate user-role database
USER ROLE ASSIGNMENT (Cont’d)

CERTIFICATE SERVER

- authenticate client
- retrieve client’s role information from user-role database
- issue certificate with client’s role information
X.509 CERTIFICATE

Serial number: seu89084jdy
Validity: 01011999 - 01012000

Subject/Name/Organization:
Common Name = Gail J. Ahn
Organization Unit = staff

Public key:
1e354276ssafatew76585098327
djkfh9974-72ks78610092wef3

Signed By: List, GMU
kljsuyto[09874875919dj28f]
4djs0475-28ejd7-18re0875

Certificate Authority

CERTIFICATE ISSUE

1. Client Certificate Request
2-3. Challenge-Response based on Password
4-5. Retrieving Role Information of a User
6-7. Creating Certificate Enrollment Form and Public-key Embedded
8-9. Issuing Client Certificate
10. Downloading Client Certificate
11. Logging Certificate Information
CERTIFICATE
AUTHORIZATION OVER SSL

client

server certificate

certificate

Role authorization

SSL connection

Task Server

REVERSE PROXYING
(MINIMAL CHANGES IN SERVER SIDE)

client

Proxy Server

Task Server

SSL connection

Request resource

Send modified request

Forward resource

Send resource

task.html

http://a.com/task.html

http://b.com/task.html