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>

© Ravi Sandhu 2000
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

<table>
<thead>
<tr>
<th>Name_Cookie:</th>
<th>Domain</th>
<th>Flags</th>
<th>Path</th>
<th>Cookie_Name</th>
<th>Cookie_Value</th>
<th>Secure</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Name_Cookie</td>
<td>Alice*</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Role_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Role_Cookie</td>
<td>manager*</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Life_Cookie</td>
<td>acme.com</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_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>Key_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Key_Cookie</td>
<td>encrypted_key*</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Seal_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Seal_Cookie</td>
<td>Seal of Cookie*</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
</tbody>
</table>

* 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
A Set of Secure Cookies

How to Use Secure Cookies

© Ravi Sandhu 2000
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>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>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>Pswd_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/ Pswd_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>/ Kerberos_Ticket</td>
<td>{Alice, K Cs)Ks</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Sign_Cookie</td>
<td>acme.com</td>
<td>TRUE</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 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
Kerberos-Based Authentication by Secure Cookies

1. Request TGT
2. TGT_Cookie = [TGT, S]Kc
3. TGT_Cookie = TGT
4. KT_Cookie = Tc5
   KC_Cookie = {Kc5, Bob}S
5. KT_Cookie = Tc5
6. TSS_Cookie

TGT = [S, Alice]Kse
TSK_Cookie = [timestamp]S, Alice, Bob
Tc5 = [Alice, Kc5]Ks (ticket to Bob)
TSS_Cookie = [timestamp]Kc5
TSS_Cookie = [timestamp+1]Kc5

Server (Bob)

© Ravi Sandhu 2000

Secure Cookies for Pay-Per-Access

- Name_Cookie
- Ticket_Cookie
- Life_Cookie
- Pwd_Cookie
- Key_Cookie
- Seal_Cookie

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

© Ravi Sandhu 2000
## Secure Cookies for RBAC

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Flag</th>
<th>Path</th>
<th>Cookie_Name</th>
<th>Cookie_Value</th>
<th>Expires</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name_Cookie</td>
<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>Role_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Role</td>
<td>Manager</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>Life_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Life_Cookie</td>
<td></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>Encrypted_Password*</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
<tr>
<td>IP_Cookie</td>
<td>acme.com</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_Cookie</td>
<td>acme.com</td>
<td>TRUE</td>
<td>/</td>
<td>Seal_Cookie</td>
<td>Digital_Signature</td>
<td>FALSE</td>
<td>12/31/99</td>
</tr>
</tbody>
</table>

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

---

## RBAC on the Web by Secure Cookies

![Diagram](attachment:image_url)

© Ravi Sandhu 2000
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)
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.
A Smart Certificate

Certificate Content:

Certificate:

Data:

Version: v3 [OpenSSL]
Serial Number: 0 (data)
Signature Algorithm: MD5 with RSA Encryption

Validity:

Not Before: Sun May 20 17:53:31 1999
Not After: Wed May 20 17:53:31 1999

Subject Public Key Info:

Algorithm: RSA 1024-bit RSA Encryption

Public Key:


Public Exponent: 65537 (0x10001)

Extensions:

Identifier: Certificate Type
Critical: False
Value: X509v3
Identifier: Subject Key Identifier
Critical: False
00:48

Signature:

Algorithm: MD5 with RSA Encryption
Signature:

© Ravi Sandhu 2000

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

© Ravi Sandhu 2000
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

- Role server
- User-role assignment
- Role-hierarchy
- User-role DB
- Certificate server

NRL design tool

- Role-hierarchy
- Role-task assignment
- CORBA IIOP
- WF system

ARCHITECTURES

- ♠ USER-PULL STYLE
- ♠ SERVER-PULL STYLE
USER-PULL STYLE

Role Server

WF design tool

WF system

SERVER-PULL STYLE

Role Server

WF design tool

WF system
NRL (Naval Research Lab.)
DESIGN TOOL

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

Platform: Windows NT, JDK1.2
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
2.1 get client certificate
2.2 retrieve role information
2.3 check authorization status
3. read resource
4. display resource
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

© Ravi Sandhu 2000
X.509 CERTIFICATE

Serial number: sou89084jdys
Validity: 01011999 - 01012000
Subject/Name/Organization
Common Name = Gail J. Ahn
Organization Unit = staff
Public key:
1e354276sofatew76585098327
djdkhe9974-72ks78610092wef3
Singed By: List, GMU
kljsuytoj09874875919jdj284jdso475-28ejd7-18re0875

Public Key

Private Key

Role Information

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

Client
Certificate Server
Certificate Engine
Log
User-Role Database

© Ravi Sandhu 2000
CERTIFICATE AUTHORIZATION OVER SSL

client

server certificate

client 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://b.com/task.html

http://a.com/task.html
FINAL SCENARIO

client

Certificate Server

Role Server

Task Server

Proxy Server

Step 1

Step 4

Step 2

Step 3

IP checking

Step 6

SSL

Step 5