Summary From the Last Lecture

- **Authentication**
  - Something you know (password)
  - Something you have (smartcard)
  - Something about you (iris scan)

- **Password authentication**
  - To protect from server compromise, store one-way hash at the server
  - To protect from dictionary attack use random, long salt
  - To protect from sniffing use Lamport hash or use asymmetric crypto instead of a password

- **Single sign-on**
How Cookies Work

- Placed into browser cache by servers to store state about this particular user
  - Contain any information that server wants to remember about the user as name/value pairs
  - May contain expiration time
  - May persist across browser instances
- Returned to server in clear on new access
- Only those cookies created for the server’s domain are sent to the server
  - May not be created by this server
- Usually used for persistent sign in, shopping cart, user preferences
Cookies for Authentication

- User logs in using her user/pass
  - Server sets a cookie with some info – username, password, session ID …
  - Any future accesses return this info to the server who uses it for authentication (equivalent to user/pass)
  - Once user signs out the cookie is deleted and the session closed at the server

- Problems
  - Cookies can be sniffed, remain on the browser because user did not sign out, be stolen by cross-site scripting or via DNS poisoning

- Solutions:
  - Send cookies over SSL, use timed cookies, secure code, bind cookies to IP address of the client, encrypt cookies …

Learn more at:
Single Sign-On

- Passport
- Liberty Alliance
- Shibboleth
Federated Identity Passport vs Liberty Alliance

- Two versions of Passport
  - Centralized and federated
- Liberty Alliance
  - Loosely federated with framework to describe authentication provided by others
Passport v1

- Goal is single sign-on
  - Solves problem of weak or repeated user/pass combinations

- Implemented via redirections
  - Users authenticate themselves to a common server, which gives them tickets
  - Similar flavor to Kerberos but different environment – many organizations

- Widely deployed by Microsoft
  - Designed to use existing technologies in servers/browsers (HTTP redirect, SSL, cookies, Javascript)
How Passport Works

- Client (browser), merchant (Web server), Passport login server
- Passport server maintains authentication info for client
  - Gives merchant access when permitted by client
- Divides client data into profile (address) and wallet (credit card)

How Passport Works

Token = 3DES encrypted authentication info using key merchant shares with passport server
Also set cookie at browser
Some Problems with Passport

- User interface is confusing and may misrepresent the reality
- Weak keys may be used for 3DES
- Single key is used to encrypt cookies for all clients
- Cookies stay on machine, can be stolen
  - No authenticator (timestamp), like in Kerberos, enables reuse by others
- Coupling of Hotmail with Passport

Read more at http://avirubin.com/passport.html
Federated Passport

- Multiple federated identity providers
  - E.g. ISPs register own users
  - One can rely on claims made by other ID providers
- Claims
  - Emails, relationships, authorization for scenarios, ownership of private/public key pair
- Need “translators” for different claim languages
Liberty Alliance

- Design criteria was most of the issues addressed by Federated Passport, i.e. no central authority
- Use SAML (Security Association Markup Language) to describe trust across authorities, and what assertions mean from particular authorities
- Four assurance levels
  - How much we trust a given identity assertion
  - Little, some, high and very high confidence
Federated Identity – Shibboleth

- **Service Provider**
  - Browser goes to Resource Manager who uses WAYF, and user’s Attribute Requester, and decides whether to grant access.
- **“Where are you from” (WAYF) service**
  - Redirects to correct servers
- **Federation to form trusted relationships between providers**
1. User requests resource
2. I don't know you, or where you are from
3. Where are you from?
4. Redirect to IdP for your org
5. I don’t know you. Authenticate using your org’s web login
6. I know you now. Redirect to SP, with a handle for user
7. I don’t know your attributes. Ask the IdP (peer to peer)
8. Based on attribute values, allow access to resource

Source: Kathryn Huxtable  
khuxtable@ku.edu  
10 June 2005
Generic Security Services API
Moving up the Stack

- Common API for client–server authentication
- Standard interface for choosing among authentication methods
  - Once an application uses GSS–API, it can be changed to use a different authentication method easily
    - No code rewriting required
    - Dominant implementation is Kerberos
  - Some procedure calls
    - Acquire and release credentials
    - Manage security context
    - Init, accept, and process tokens (challenges)
    - Wrap and unwrap (encrypt/decrypt)
Attacks on Password Authentication

- Brute force
- Dictionary
- Guessing
- Finding elsewhere
Something You Have

- **Cards**
  - Mag stripe (= password)
  - Smart card, USB key
  - Time-varying password

- **Issues**
  - How to validate
  - How to read (i.e. infrastructure)
Something About You

◆ Biometrics
  ○ Measures some physical attribute
    • Iris scan
    • Fingerprint
    • Picture
    • Voice

◆ Issues
  ○ How to prevent spoofing
    • Suited when biometric device is trusted, not suited otherwise
Other Forms Of Authentication

- IP Address
- Caller ID (or call back)
- Past transaction information
  - Example of something you know
Multi-factor Authentication

- Require at least two of the classes we mentioned, e.g.
  - Smart card plus PIN
  - RSA SecurID plus password
  - Biometric and password
Authorization and Policy
Authorization: Two Meanings

- **Determining permission**
  - Is principal P permitted to perform action A on object U?

- **Adding permission**
  - P is permitted to perform action A on object U

- In this course, we use the first definition
Access Control

- Who is permitted to perform which actions on what objects?
- **Access Control Matrix (ACM)**
  - Columns indexed by principal
  - Rows indexed by objects
  - Elements are arrays of permissions indexed by action
- **In practice, ACMs are abstract objects**
  - Huge and sparse
  - Possibly distributed
## Example ACM

<table>
<thead>
<tr>
<th>File/User</th>
<th>Tom</th>
<th>Dick</th>
<th>Harry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Readme.txt</td>
<td>read</td>
<td>read</td>
<td>read, write</td>
</tr>
<tr>
<td>passwords</td>
<td></td>
<td></td>
<td>write</td>
</tr>
<tr>
<td>Term.exe</td>
<td>read, write, execute</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Instantiations of ACMs

- **Access Control Lists (ACLs)**
  - For each object, list principals and actions permitted on that object
  - Corresponds to rows of ACM

<table>
<thead>
<tr>
<th>File/User</th>
<th>Access Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Readme.txt</td>
<td>Tom: read, Dick: read, Harry: read, write</td>
</tr>
<tr>
<td>passwords</td>
<td>Harry: write</td>
</tr>
<tr>
<td>Term.exe</td>
<td>Tom: read, write, execute</td>
</tr>
</tbody>
</table>
Instantiations of ACMs

- **Capabilities**
  - For each principal, list objects and actions permitted for that principal
  - Corresponds to columns of ACM
- **The Unix file system is an example of...?**

<table>
<thead>
<tr>
<th>User</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Readme.txt: read, Term.exe: read, write, execute</td>
</tr>
<tr>
<td>Dick</td>
<td>Readme.txt: read</td>
</tr>
<tr>
<td>Harry</td>
<td>Readme.txt: read, write; passwords: write</td>
</tr>
</tbody>
</table>
Problems

- Permissions may need to be determined dynamically
  - Time
  - System load
  - Relationship with other objects
  - Security status of host

- Distributed nature of systems may aggravate this
  - ACLs need to be replicated or centralized
  - Capabilities don’t, but they’re harder to revoke
Types of Access Control

- **Discretionary**
  - Owners control access to objects
  - Access permissions based on identity of subject/object
  - E.g., access to health information

- **Mandatory**
  - System controls access to objects via rules
  - E.g., doctors can read files of their own patients

- **Originator-controlled**
  - Creator controls access to objects, not the owner
  - E.g., owner can listen to a song but not share it
Authorization

- Final goal of security
  - Determine whether to allow an operation
- Depends upon
  - Policy
  - Authentication
  - Other characteristics
The Role Of Policy

- Policy defines what is allowed and how the system and security mechanisms should act.
- Policy is enforced by mechanism which interprets it, e.g.
  - Firewalls
  - IDS
  - Access control lists
- Implemented as
  - Software (which must be implemented correctly and without vulnerabilities)
Policy models: Bell–LaPadula

- Focuses on controlled access to classified information and on confidentiality
  - No concern about integrity
- The model is a formal state transition model of computer security policy
  - Describes a set of access control rules which use security classification on objects and clearances for subjects
- To determine if a subject can access an object
  - Combine mandatory and discretionary AC (ACM)
  - Compare object’s classification with subject’s clearance (Top Secret, Secret, Confid., Unclass.)
  - Allow access if ACM and level check say it’s OK
Policy models: Bell–LaPadula

- Three security properties:
  - Simple Security Property – a subject at a given security level may not read an object at a higher security level (no read-up)
  - Star Property – a subject at a given security level must not write to any object at a lower security level (no write-down). Strong Star Property – only write to same level
  - The Discretionary Security Property – discretionary access control specified via an access control matrix

- Trusted subjects – no star property rule
  - Transfer info from high clearance to low clearance
Role-Based Access Control

- Ability to access objects depends on one’s role in the organization
- Roles of a user can change
  - Restrictions may limit holding multiple roles simultaneously or within a session, or over longer periods.
  - Supports separation of roles
- Maps to organization structure
Integrity Policies: Biba Model

- Like Bell–LaPadula but speaks about integrity
- Cannot write to higher-level objects
- Subject’s integrity drops if it reads a lower-level object
- A subject can only execute subjects at a lower integrity level
Paper Reports Due Next Thursday

- Due midnight
- Check the Web page for the paper template
  - Either use this template or make sure you have same pieces of information in your submission