CS 138: Security II
Today

- Secure key distribution
- Authorization
Authentication with Shared Secret

Alice

Bob

A

R_B

K_{AB}(R_B)

R_A

K_{AB}(R_A)
Shortcut

A, \( R_A \)

\( R_B, K_{AB}(R_A) \)

\( K_{AB}(R_B) \)

Alice

Bob
Trickery: Reflection Attack
Fixing the replay

Alice

A, R_A

K_{AB}(R_B,R_A)

K_{AB}(R_B^{-1})

Bob
Problem: $n^2$ key pairs!

- Alternatives
  - Share keys with a key distribution service
  - Public-key cryptography
Kerberos

• Developed at MIT in the 80’s
• Uses a Key Distribution Service (KDC)
  – Based on Needham-Shroeder key exchange
• Our description based on the “play”:
  “Designing an Authentication System: a Dialogue in Four Scenes”

http://web.mit.edu/kerberos/dialogue.html
Kerberos alpha 0

A, password, mail

K_{mail}(A)

KDC

A, K_{mail}(A)

Alice

Mail
Kerberos alpha 1

Diagram:

- Alice
- KDC
- Mail

Messages:

- A, mail
- \( K_A(\text{mail}, K_{\text{mail}}(A)) \)
- \( A, K_{\text{mail}}(A) \)
Avoiding Replays

\[ K_A(\text{mail}, K_{\text{mail}}(A, \text{exp})) \]

Alice \rightarrow KDC \rightarrow Mail

\[ A, K_{\text{mail}}(A, \text{exp}) \]
Avoiding Replays

\[ K_A(K_{AM}), K_{mail}(A, K_{AM}, exp) \]

\[ K_{AM}(A), K_{mail}(A, K_{AM}, exp) \]
Avoiding Replays

\[ K_A(K_{AM}), K_{mail}(A, K_{AM}, \exp) \]

\[ K_{AM}(A, ts), K_{mail}(A, K_{AM}, \exp) \]
Ticket granting service

A, TGS

$K_A(K_{AT}), K_{TGS}(A,K_{AT},exp)$

$K_{AT}(A,ts_1), \text{mail, } K_{TGS}(A,K_{AT},exp)$

$K_{AT}(K_{AM}), K_{mail}(A, K_{AM}, \text{exp})$

$K_{AM}(A, ts_2), K_{mail}(A, K_{AM}, \text{exp})$
Authenticating the server

\[ K_A(K_{AT}), K_{TGS}(A,K_{AT},\text{exp}) \]

\[ K_{AT}(A,ts_1), \text{mail, } K_{TGS}(A,K_{AT},\text{exp}) \]

\[ K_{AT}(K_{AM}), K_{\text{mail}}(A, K_{AM}, \text{exp}) \]

\[ K_{AM}(A, ts_2), K_{\text{mail}}(A, K_{AM}, \text{exp}) \]

\[ K_{AM}(ts_2+1) \]
Cross-Realm Authentication

Realm X

Realm Y

Client

Application Server

Authentication Server

Ticket-Granting Server

Kerberos Key-Distribution Server
Transitive Trust

Realm A

Realm B

Realm Z

Client

Application Server

Client
Hierarchical Trust
Getting Authorized

Send me a copy of a journal

Are you a paid member?
Getting Authorized
Getting Authorized

I’m a Brown student.

Prove it.
Getting Authorized

My IP address starts 128.148.

Good enough for me.
Getting Authorized

Thank you

Hacks ’R’ Us
Getting Authorized

I need a hack for 138.

Prove you are a 138 student.
Enter Shibboleth

Hacks ‘R’ Us
Using Shibboleth

• Student
  – logs in to Brown, gets credentials

• Hacks ’r’ Us
  – responds to client requests with an authentication request
    - indicates what it requires (e.g., CS138 student status)

• Identity provider
  – contacted by student’s browser
  – given student’s credentials, returns desired student attributes (CS 138 student)
Shibboleth

- Separates the federation from the authentication
  - Individual IdP’s can do what they want
  - Federation makes it more scalable
Diffie-Hellman Key Exchange

- Different model of the world: How to generate keys between two people, securely, no trusted party, even if someone is listening in.

This is cool. But: Vulnerable to man-in-the-middle attack. Attacker pair-wise negotiates keys with each of A and B and decrypts traffic in the middle. No authentication...
Authorization

• Is the requestor permitted to perform the requested operation?
• Does this require knowledge of who the requestor is?
An analogy

• Alice wants a safe deposit box in a bank
• Two options:
  – Bank maintains a list of who can access the box
  – Bank gives Alice a key (or a combination)
• What are the pros and cons?
ACL-Based Authorization

Authenticated Client → Reference Monitor → Service
Capability-Based Authentication

Anonymous Client → capability → Service
Making ACLs Work

• Client provides credentials
  – privilege attribute certificate (PAC)
    - certificate listing client’s credentials
      • e.g., user name, groups, etc.
• Client requests a particular operation
• Server’s reference monitor looks up credentials and request in ACL
  – returns permit/deny decision
Privilege Server

- Extend Kerberos into *Privilege Server*
  - maintains user and group database
  - prepares PACs
    - includes them in ticket
    - application-server ticket informs server of all of client’s credentials
Impersonation

Authenticated Client → Reference Monitor → Service

Print Server

File Server

allow twd w
...

allow twd r
...

Reference Monitor → Service
Impersonation using Privilege Server

- Client requests print-server ticket from privilege server
  - asks it to mark PAC “permit impersonation”
- Client sends ticket to print server
- Print server requests file-server ticket from privilege server
  - includes client’s print-server ticket
  - privilege server provides file-server ticket containing original client’s PAC
    - print server impersonates client
Impersonation Problems

Authenticated Client

Print Server
  \[\text{Reference Monitor} \rightarrow \text{Service}\]

Cash Server
  \[\text{Reference Monitor} \rightarrow \text{Service}\]

File Server
  \[\text{Reference Monitor} \rightarrow \text{Service}\]
Delegation

Authenticated Client

Reference Monitor

Print Server

Service

Cash Server

Reference Monitor

allow twd rw

File Server

Reference Monitor

allow twd_delegates rw

Service

Service
How It’s Done

• Client requests print-server ticket with delegation permitted
  – privilege server constructs ticket with client’s PAC so marked
• Client presents ticket to print server
• Print server requests delegated file-server ticket from privilege server
  – privilege server returns ticket with both original client’s and print-server’s PACs
• Print server presents ticket to file server
  – file server checks delegate entries in ACL
Capabilities

• A capability is both a reference and an access right to a particular resource
ACLs vs. C-Lists

Rob's Process

File X

Rob: rw
Chris: r
ACL

File Y

Rob: r
Chris: rw
ACL

Chris's Process

Rob's Process

C-List

rw
r

Chris's Process

C-List

r
rw
More General View

• Subjects and resources are objects (in the OO sense)

Diagram:
- Object A
  - read
- Object B
  - append
- Object C
Copying Capabilities (2)

Object A

write cap
read

Object B

read

Object C
“Directories”

Object A

read cap

Object B

read cap

Directory

read
write
append

Object X

Object Y

Object Z
Least Privilege (1)

- Login Process
  - read cap
  - write cap

- Directory
  - read
  - write
  - read

- Public Data
- System File
- Credit Card Info
Least Privilege (2)

Login Process → Directory
- read cap
- write cap

Suspect Code → Directory
- read

Public Data → System File → Credit Card Info
- read
- write
- read
An analogy

<table>
<thead>
<tr>
<th></th>
<th>ACL (List)</th>
<th>Capability (Key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Bank must check list</td>
<td>Bank not involved</td>
</tr>
<tr>
<td>Forging access</td>
<td>Bank must secure list</td>
<td>Can’t be forged</td>
</tr>
<tr>
<td>Adding a new person</td>
<td>Owner visits bank</td>
<td>Copy key</td>
</tr>
<tr>
<td>Delegation</td>
<td>Friend can’t delegate</td>
<td>Friend can give key</td>
</tr>
<tr>
<td>Revocation</td>
<td>Owner can remove ex</td>
<td>Harder</td>
</tr>
</tbody>
</table>

- Sharing online album
  - Authorize specific users
  - Share by secret URL
ACLs vs. Capabilities

- **ACLs**
  - Authentication
    - Reference monitor involved
  - specifying access rights
    - easy
  - least privilege
    - hard
  - delegation
    - Awkward
  - Revocation
    - easy

- **Capabilities**
  - Authentication
    - No one involved
  - specifying access rights
    - awkward
  - least privilege
    - easy
  - delegation
    - Easy
  - Revocation
    - hard
Capabilities in Amoeba

Object reference

- server port: 48 bits
- object: 24 bits
- rights: 8 bits
- check: 48 bits

Copy kept on server
Generating Restricted Capabilities

server port  |  object  |  11111111  |  C

new rights 00000001

Xor

One-way Function

server port  |  object  |  00000001  |  f(C⊕00000001)