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

Mallory – Bob

A, R_M

R_B, K_{AB}(R_M)

A, R_B

R_{B2}, K_{AB}(R_B)

K_{AB}(R_B)
Fixing the replay

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

Alice

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”
  
  [Link: http://web.mit.edu/kerberos/dialogue.html]
Kerberos alpha 0

A, password, mail

K_{mail}(A)

KDC

Alice

A, K_{mail}(A)

Mail
Kerberos alpha 1

A, mail

$K_A(\text{mail}, K_{\text{mail}}(A))$

KDC

A, $K_{\text{mail}}(A)$

Mail
Avoiding Replays

$$K_A(mail, K_{mail}(A, exp))$$

$$A, K_{mail}(A, exp)$$

Alice

KDC

Mail
Avoiding Replays

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

A, mail

KDC

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

Alice

Mail
Avoiding Replays

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

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

A, TGS

K_A(K_AT), K_TGS(A,K_AT,exp)

K_AT(A,ts_1), mail, K_TGS(A,K_AT,exp)

K_AT(K_AM), K_mail(A, K_AM, exp))

K_AM(A, ts_2), K_mail(A, K_AM, exp)

KDC

TGS

Alice

Mail
Authenticating the server

KDC

A, TGS

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

TGS

K_{AT}(A,\text{ts}_1), \text{mail}, K_{TGS}(A,K_{AT},\text{exp})

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

Mail

K_{AM}(A, \text{ts}_2), K_{mail}(A, K_{AM}, \text{exp})

K_{AM}(\text{ts}_2+1)
Cross-Realm Authentication

Realm X

Realm Y

Client

Application Server

Client

Application Server
Transitive Trust

Realm A

Realm B

Realm Z

Client

Application Server

Client

Client
Hierarchical Trust

/acme.com
/west_coast
/manufac
/east_coast
/R&D
/osf.org
/RI
/college.edu
/CS
/admin
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.

Hacks ’R’ Us
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
Capability-Based Authentication

Anonymous Client

Service

capability
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

Print Server

Reference Monitor

Service

File Server

Reference Monitor

Service

allow twd w
...

allow twd r
...

Copyright © 2012 Thomas W. Doeppner, Rodrigo Fonseca. All rights reserved.
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

Reference Monitor

Service

Reference Monitor

Service

Cash Server

Reference Monitor

Service

Reference Monitor

Service

File Server
Delegation

Authenticated Client

Print Server
  Reference Monitor
  Service

Cash Server
  Reference Monitor
  Service

File Server
  Reference Monitor
  Service

allow twd_rw

allow twd_delegates_rw
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

rw
r
C-List

CHRIS’s Process

r
rw
C-List
More General View

- Subjects and resources are *objects* (in the OO sense)

![Diagram showing object relationships]

Object A \(\rightarrow\) Object B \(\rightarrow\) Object C

read \(\rightarrow\) append
Copy Capabilities (1)

Object A

write cap
read

Object B

Object C
Copying Capabilities (2)
“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
  - read cap
  - write cap
- Directory
  - read
  - write
  - read
- Suspect Code
  - read
- Public Data
- System File
- Credit Card Info
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

- Server port: 48 bits
- Object: 24 bits
- Rights: 8 bits
- Check: 48 bits

Object reference

Copy kept on server
Generating Restricted Capabilities

server port | object | 11111111 | C

new rights

Xor

One-way Function

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