CS 138: Security Continued
Time Synchronization
Today

• Review of secure key distribution
• Authorization
Authentication with Shared Secret

Alice

\[ R_B \]

\[ K_{AB}(R_B) \]

\[ R_A \]

\[ K_{AB}(R_A) \]

Bob
Shortcut

A, $R_A$

$R_B, K_{AB}(R_A)$

$K_{AB}(R_B)$

Alice  ↔  Bob
Trickery: Reflection Attack

Mallory

A, \( R_M \)

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

Bob

A, \( R_B \)

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

\( K_{AB}(R_B) \)
Fixing the replay

Bob

Alice

A, R_A

R_B, K_{AB}(R_A)

K_{AB}(R_B^{-1})
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 Key Distribution Service
  - 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

$A, \text{password, mail}$

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

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

KDC

Alice

Mail
Kerberos alpha 1

A, mail

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

KDC

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

Mail

Alice
Avoiding Replays

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

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

$\text{K}_A(\text{K}_{\text{AM}}), \text{K}_{\text{mail}}(\text{A}, \text{K}_{\text{AM}}, \text{exp})$

Alice

KDC

Mail

$\text{K}_{\text{AM}}(\text{A}), \text{K}_{\text{mail}}(\text{A}, \text{K}_{\text{AM}}, \text{exp})$
Avoiding Replays

Alice

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

KDC

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

Mail
Ticket granting service

KDC

A, TGS

K_A(K_AT), K_{TGS}(A,K_AT,exp)

TGS

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)

Alice

Mail
Authenticating the server

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)

K_AM(ts_2+1)

KDC

TGS

Alice

Mail
Cross-Realm Authentication

Realm X

Realm Y

Client

Application Server
Transitive Trust

Realm A

Realm B

Realm Z

Client

Application Server

Client
Hierarchical Trust

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

Red lines indicate trust relationships.
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

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 → Reference Monitor → Service

Print Server

Reference Monitor → Service

File Server

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

Reference Monitor

Service

Print Server

Cash Server

Reference Monitor

Service

File Server

Reference Monitor

Service
Delegation

Authenticated Client → Reference Monitor → Service

Cash Server

Reference Monitor → Service

Print Server

Reference Monitor → Service

File Server

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

File X

rw
r
C-List

File Y

r
rw
C-List
More General View

• Subjects and resources are *objects* (in the OO sense)
Copying Capabilities (1)

Object A

write cap
read

Object B

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)
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

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

Object reference

Copy kept on server
Generating Restricted Capabilities

server port  object  \( \text{11111111} \)  \( C \)

\( C \oplus \text{00000001} \)

\( \text{00000001} \)

Xor

One-way Function

server port  object  \( \text{00000001} \)  \( f(C \oplus \text{00000001}) \)
Getting Authorized

Send me a copy of a journal

Are you a paid member?
Getting Authorized

Thank you

ACM DL DIGITAL LIBRARY

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