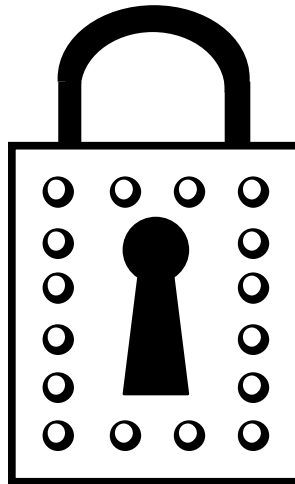


Access Control List (ACL)



License

Copyright © 2008 Ciaran McHale.

Permission is hereby granted, free of charge, to any person obtaining a copy of this training course and associated documentation files (the "Training Course"), to deal in the Training Course without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Training Course, and to permit persons to whom the Training Course is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Training Course.

THE TRAINING COURSE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE TRAINING COURSE OR THE USE OR OTHER DEALINGS IN THE TRAINING COURSE.

Access control list

- An *access control list* (ACL) is an *authorization* mechanism
 - Specifies what access permissions different users have for a resource
 - Examples of resources: a file, a printer, operations on an object in a server application
- Many systems provide ACLs, but the mechanisms vary
- Example using a pseudo-code syntax for an RPC system:

user Fred can execute:

SessionManager.login
SessionManager.logout
Session.*

<interface>.<operation>
“*” matches all operation names

user Mary can execute:

...

Role-based access control

- Problem: some systems have thousands of users:
 - Tedious and error prone to specify ACLs for each user individually
- Solution: role-based access control (RBAC):
 - Assign each user to one or more roles
 - Then write ACLs in terms of roles rather than individual users
- Example using a pseudo-code syntax for an RPC system:

```
user Fred belongs to employee, manager;  
user Mary belongs to employee;  
user Sam  belongs to customer;  
...  
role employee can execute: ...  
role manager can execute: ...
```

Role-based access control (cont')

- Benefit of role-based access control:
 - There may be thousands or even millions of users
 - But usually only a very small number of roles
 - So ACL maintenance is easy

Prerequisites for authorization

- A prerequisite for access control lists (or any other authorization mechanism) is *authorization*
 - No point in specifying what user Fred can do if we cannot verify whether or not a user *is* Fred
- Question: can a system provide authentication without (the overhead of) secure communications (such as SSL/TLS)?
- Answer:
 - Probably not, because...
 - Authentication is often done via username and password
 - If usernames and passwords are transmitted without encryption then a hacker can easily capture them

Summary

- Access control lists (ACLs) are widely used for authorization
- Can be tedious to specify an ACL for each of 1000's of users
 - Better to map many users to a small number of *roles*
 - Then define ACLs for the roles
 - This is called role-based access control (RBAC)
- Authentication is a prerequisite for authorization
 - And encryption is a prerequisite for authentication
(so people cannot snoop on usernames and passwords)