First half

For single-

\[ \text{read\_block} \]
\[ \text{write\_block} \]

FUSE

```
getcttlo("/home/file.txt")
```

Look at through root directory, find home and associated block, in that find file: file.text: attributes

\* -> Protection 
\* -> Security
restricted access
- privacy
- non-interference
- integrity
- identification

* Goals:

Shared Access { privacy
- non-interference

* Attacks
- Internal
- External \&\ such as internet

* Robustness
- to make system more reliable

* Confidentiality

\[\rightarrow\] Data \& you want to restrict\[\rightarrow\] actions \&\ not be watched or manipulated by others

* Integrity: data operation

\[\rightarrow\] Trojan (modified command)
Availability

X -→ read, open,

actor

operation

objects

actor (user ID) → object with processes, objects

action

permission decision

L → (actor, action, object) → allow

permissions:

object → { (user, action) }

coherent with each user can perform what action on it

(other ways to organize it)

capabilities

user → { (object, action) }
Second half

Permissions:

- Process: 
  - UID
  - GID

- Object: 
  - Owner
  - Group

User, Group, Other

- 16-bit numbers

Systems to have separate permission to delete the file

Alternative approach is ACLs (Access Control List)

- Set of rules identified for particular objects

<identity> <action> allow/deny

- User/group
- User: allow
- Group: allow
- Other: allow

Negative point

Variable length structure
Advo - Flexible control

Access matrix:

<table>
<thead>
<tr>
<th></th>
<th>File 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>User 2</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>User 3</td>
<td>yes</td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>no</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

File 1: owner U1
      groups (1, 2)

File 2: owner U2
      group (1, 3, 3)
      rwo - - -

ACL is sufficient to express above matrix

Example:

Owner: rwx
User 2: rwx
User 3: r-
User 4: r-

Others: - - -
CHOWN - change owner
KILL - kill process
MKV00 -
NET-RAW
REBOOT
Load module - load module to kernel
(Not in non-root process)
Don't need to allow web server root dir
 Load module to kernel

Authentication:
  - login (text)
  - graphical login
    e.g. /etc/xdm
    xdm, kdm, gdm
    login, login1
    gdm

Something you:
  know - password
  have - token (like RSA key)
  are - confused
  do - biometrics

Implementation:
How could you check password? - std:

1. Store password in data file, search it
2. Hashed
   \[ abc/123 \rightarrow \text{F} \rightarrow x73\text{E7} \]
   one-way function.

Cracking - dictionary attack can be used to hack this function

- \[ abc/123 <\text{nonce} > \rightarrow \text{Hash(Password + nonce)} \]
  nonce
  Now you have nonce
  and hash value of password file

Password over a network:

1. public key encryption
2. challenge/authenticate protocol
name → system

plaintext password file

Hash (performance)