Notes for the lecture conducted on 12/3 on the topic – Access Control

Access controls are of two types:

- Mandatory Access Controls
  - The security policy (mentioned below) is enforced by the system
- Discretionary Access Controls
  - The security policy is decided by the user.

There are two aspects to Access Controls:-

- Policy
- Mechanism

An example of policy would be that a web browser is allowed to talk to a web server who in turn talks to the database server. However, the web browser is not allowed to talk to the database server directly.

Whenever it is possible it is wise to keep policy and mechanism separate.

An access control mechanism is precisely as the same suggests i.e. a mechanism to restrict or allow access.

A simple way to implement access control is by the use of an access control matrix.

It consists of users (subjects) and the permissions (objects) they possess.

<table>
<thead>
<tr>
<th></th>
<th>etc/passwd</th>
<th>Finals.tex</th>
<th>~student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Student</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Admin</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The above figure represents an access control matrix that stores binary information. The columns represent subjects and rows represent objects. A binary ‘1’ implies that the particular subject has access to the object. A ‘0’ implies that the particular subject does not have access to the object.

A less restrictive form of storing values in an access control matrix would be to specify the permissions possessed by subjects rather than just binary values. This is shown in the matrix below:

<table>
<thead>
<tr>
<th></th>
<th>/etc/shadow</th>
<th>Finals.tex</th>
<th>~student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor</td>
<td></td>
<td>Read/Write</td>
<td></td>
</tr>
<tr>
<td>student</td>
<td></td>
<td></td>
<td>Ins/rm/list/query</td>
</tr>
<tr>
<td>admin</td>
<td></td>
<td>append</td>
<td></td>
</tr>
</tbody>
</table>

An even more flexible method to store values would be to specify ownership. This way access control can be enforced on the basis of who owns that particular file in addition to what permissions he or she has.

Every row in an access control matrix represents a capability.

A column represents an Access Control List (ACL). It is typically stored along with the object and lists users and permissions.

The associated semantics may be order dependent or order independent.

First-match semantics are order dependent whereas any-allowing-match semantics are order independent.

Order independent semantics are preferred as they are more intuitive.
The Bell Lapadula access control is based on tagging files with the security clearance that a person is supposed to have at a certain level. Objects are tagged with the clearance level required to access them. Subjects are tagged with the level of clearance they possess.

A tag comprises of the level of clearance and a compartment set. This can be represented as a tuple as $T_s = (L_s, C_s)$ for subjects.

For objects, tags are represented as $T_o = (L_o, C_o)$. Files can be accesses only if $L_s$ is greater than or equal to $L_o$ AND $C_s$ is a superset of $C_o$. Here, $T$, $L$ and $C$ represent tag, level and compartment respectively and the subscripts $s$ and $o$ represent subjects and objects respectively.