**Ways of handling security issues**

**Full disclosure:**
- announce the bugs to the public
- forces the developer to take quick action
- safe to do when there are alternatives

Inform the developers first and then disclose to public
- avoids unwanted panic among the public
- reduces further exploitation of the problem by the bad guys

**Trust**

Trust is about dependence.

Problem: Trust is transitive

Therefore, always minimize trust.

**To reduce trust**
- write the code yourself
- check code with multiple vendors
- cross-check (using different compiler) / replication (RAID)
- scramble the code, so that pattern matching fails and it becomes difficult to infect
- make the code simple, so that you can check with multiple people.

**Core security system design principles**

1. **Keep it simple**
   - Have a small TCB (Trust Computing Base)

   From Wikipedia: The **trusted computing base** (TCB) of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs or vulnerabilities occurring inside the TCB might jeopardize the security properties of the entire system
   - Economy of mechanism

   100 lines of code is more secure than 1000 lines of code
2. **Fail safe defaults**
   When a system fails, let it fail in secure ways.

<table>
<thead>
<tr>
<th>Default Deny</th>
<th>Default allow</th>
</tr>
</thead>
<tbody>
<tr>
<td>More secure</td>
<td>Less secure</td>
</tr>
<tr>
<td>Users will complain</td>
<td>Users will not complain</td>
</tr>
<tr>
<td>Fails loudly</td>
<td>Fails silently</td>
</tr>
<tr>
<td>Requires list of all bad things that can happen</td>
<td></td>
</tr>
</tbody>
</table>

3. **Least privilege**
   When a small program has to be integrated into a larger one, give it least amount of access/privilege to do the same
   e.g. Amazon has different web servers accessing the main database

4. **Complete mediation**
   - If you are restricting access to something, protect every possible way of accessing it (all vulnerable points).
   - Best way is to minimize the number of access paths.

5. **Separation of privilege**
   - Split the access right among 2 people, so it requires both of them to authorize action.
   - Purpose: enables double check
   - E.g. 2 people should turn 2 different keys at the same time to launch nuclear bomb
   - E.g. 2 compilers are used and both should be correct

Example:

If the mail server has a worm, it may get passed on to its clients.
So do not have one program do both receiving and sending functions.
For worm removal, do separation of privilege.
6. **Least shared mechanism**
   - Advantage of not sharing code is limited damage and helps in reducing trust.
   - Nowadays things are moving towards less data sharing and more code sharing.
   - Least shared state
     - For password database and product database, keep the state separate even though it is sharing the same software (oracle) to implement.

7. **Psychological acceptability**
   - Firewalls in corporate networks try to implement lot of security, but employees find a way out other links to access blocked sites.
   - This violates ‘Complete mediation’.