Computers II Lesson 7

7.0 Security Design

Security engineering focuses on how to develop and maintain software systems that can resist malicious attacks intended to damage a computer-based system or its data.

Security threats can be threats to the:

- Confidentiality of a system or its data
- Integrity of a system or its data
- Availability of a system or its data

Designing a system architecture that maintains security, you need to consider these two issues:

1. Protection—how should the system be organized so that critical assets can be protected against external attack?

2. Distribution—how should system assets be distributed so that the effects of a successful attack are minimized?

There are exact rules on how to achieve system security.

Different types of systems require different technical measures to achieve a level of security that is acceptable to the system owner.

- For example, in a bank, users are likely to accept a higher level of security, and hence more intrusive security procedures than in a university.

However, there are general guidelines that have wide applicability when designing system security solutions, which encapsulate good design practice for secure systems engineering.

These general design guidelines for security have two principal uses:

1. They help raise awareness of security issues in a software engineering team.

Software engineers often focus on the short-term goal of getting the software working and delivered to customers. It is easy for them to overlook security issues. Knowledge of these guidelines can mean that security issues are considered when software design decisions are made.

2. They can be used as a review checklist that can be used in the system validation process. From the high-level guidelines discussed here, more specific questions can be derived that explore how security has been engineered into a system.

Security guidelines
1 Base security decisions on an explicit security policy
2 Avoid a single point of failure
3 Fail securely
4 Balance security and usability
5 Log user actions
6 Use redundancy and diversity to reduce risk
7 Validate all inputs
8 Compartmentalize your assets
9 Design for deployment
10 Design for recoverability

Guideline 1: Base security decisions on an explicit security policy

A security policy defines the 'what' of security rather than the 'how', so the policy should not define the mechanisms to be used to provide and enforce security.

- For example, say you are designing an access control system for a hospital. Their security policy may state that only accredited clinical staff may modify electronic patient records. Your system therefore has to include mechanisms that check the accreditation of anyone attempting to modify the system and that reject modifications from people who are not accredited.

Guideline 2: Avoid a single point of failure

In any critical system, it is good design practice to try to avoid a single point of failure. This means that a single failure in part of the system should not result in an overall systems failure.

In security terms, this means that you should not rely on a single mechanism to ensure security; rather you should employ several different techniques. This is sometimes called 'defense in depth'

- For example, if you use a password to authenticate users to a system, you might also include a challenge/response authentication mechanism where users have to pre-register questions and answers with the system.

Guideline 3: Fail securely

System failures are inevitable in all systems.

When the system fails, you should not use fallback procedures that are less secure than the system itself. Nor should system failure mean that an attacker can access data that would not normally be allowed.

Guideline 4: Balance security and usability

The demands of security and usability are often contradictory. To make a system secure, you have to introduce checks that users are authorized to use the system and that they are acting in accordance with security policies. All of these inevitably make demands on users—they may have to remember login names and passwords, only use the system from certain computers, and so on. These mean that it takes users more time to get started with the system and use it effectively. As you add security features to a system, it is inevitable that it will become less usable.

- For example, if you require users to input multiple passwords or to change their passwords to impossible-to-remember character strings, they will

simply write down these passwords. An attacker (especially an insider) may then be able to find the passwords that have been written down and gain access to the system.

Guideline 5: Log user actions

You should always maintain a log of user actions. This log should, at least, record who did what, the assets used, and the time and date of the action. It also acts as a deterrent to insider attacks. If people know that their actions are being logged, then they are less likely to do unauthorized things. However, a technically skilled insider can also access and change the log.

Guideline 6: Use redundancy and diversity to reduce risk

Redundancy means that you maintain more than one version of software or data in a system.

Diversity, when applied to software, means that the different versions should not rely on the same platform or be implemented using the same technologies.

Guideline 7: Validate all inputs

A common attack on a system involves providing the system with unexpected inputs that cause it to behave in an unanticipated way.

These may simply cause a system crash, resulting in a loss of service, or the inputs could be made up of malicious code that is executed by the system.

Guideline 8: Compartmentalize your assets

Compartmentalizing means that you should not provide all-or-nothing access to information in a system.

You should organize the information in a system into compartments. Users should only have access to the information that they need, rather than to all of the information in a system.

This means that the effects of an attack may be contained. Some information may

be lost or damaged but it is unlikely that all of the information in the system will be affected

- For example, in the patient information system, you should design the system so that at any one clinic, the clinic staff normally only has access to the records of patients that have an appointment at that clinic. They should not normally have access to all patient records in the system. Not only does this limit the potential loss from insider attacks, it also means that if an intruder steals their credentials, then the amount of damage that they can cause is limited.

Guideline 9: Design for deployment

Many security problems arise because the system is not configured correctly when it is deployed in its operational environment. You should always design your system so that facilities are included to simplify deployment in the customer's environment and to check for potential configuration errors and omissions in the deployed system.

Guideline 10: Design for recoverability

You should always design your system with the assumption that a security failure could occur. Therefore, you should think about how to recover from possible failures and restore the system to a secure operational state.

- For example, you may include a backup authentication system in case your password authentication is compromised.

Attack	Resistance	Recognition	Recovery
Unauthorized user places malicious orders	Require a dealing password that is different from the login password to place orders.	Send copy of order by e-mail to authorized user with contact phone number (so that they can detect malicious orders). Maintain user's order history and check for unusual trading patterns.	Provide mechanism to automatically 'undo' trades and restore user accounts. Refund users for losses that are due to malicious trading. Insure against consequential losses.