SY308: Security Fundamental Principles

Calendar	Policy

Security Principles

Fundamental Security Design Principles

Saltzer and Schroder

They presented the following security design principles [SS75].

1. **Economy of mechanism**

   > Keep it simple.

   This is a good guideline whenever a problem needs to be solved, and it often reflects the quality of solution.

   - The simpler the systems, the easier they are to understand.
   - The simpler the systems, the less likely they are to contain flaws.

2. **Fail-safe Default**

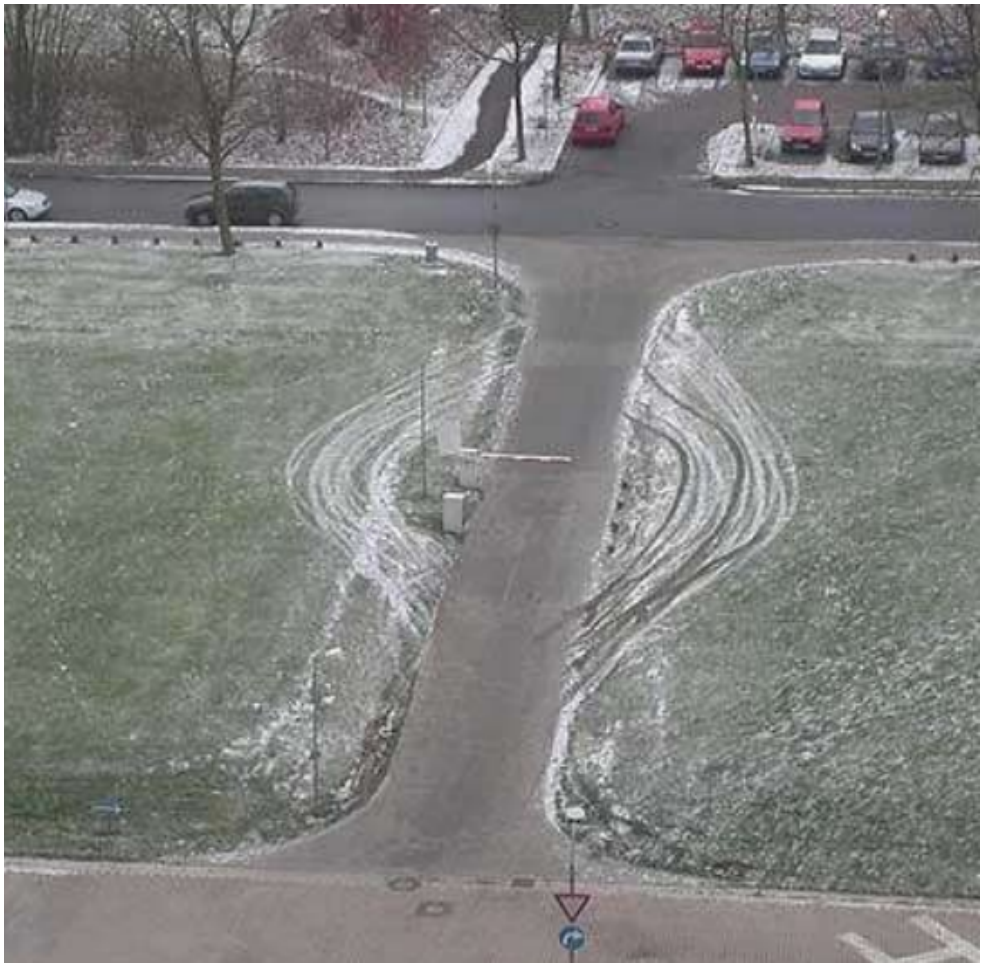
   > Unless an entity is given explicit access to an object, it should be denied access to that object.

   That is, the default situation is lack of access. This is often called a *whitelist approach*. With this approach, mistakes in the mechanism would be probably denying access by authorized entities (but not in the whitelist), which can be quickly detected. Most file access systems work on this principle and virtually all protected services on client/server systems work this way.

   The opposite variant is the *blacklist approach*; that is, permission is granted unless explicitly denied. With this approach, mistakes in mechanism would probably be allowing access by unauthorized entities (but not in the blacklist), which may long go unnoticed.

3. **Complete Mediation**

   > Every access must be monitored and controlled.



An access control mechanism must encompass all relevant objects and must be operational in any state the system can possibly enter. Care should be taken to ensure that the access control mechanism cannot be circumvented. For example:

- Sensitive information should be protected even during transit and in storage, which often requires data to be encrypted for complete mediation. If the information is unencrypted, the adversary may boot a different OS to circumvent file-system-based access control, or sniff traffic to circumvent access control imposed by a web application.
- One should be wary of performance improvements techniques that **saves the results of previous authorization checks**, since **permissions can change over time**. For example, it can be risky if permissions are checked the first time a program requests access to a file, but subsequent accesses to the same file are not checked again while the application is still running.

4. **Open Design**

> The security of a system should not depend on the obscurity of its protection mechanism.

Security should rely **only on keeping cryptographic keys or passwords secret**; in particular, the system should remain secure even if the security architecture and design of a system are publicly available.

This decoupling of protection of mechanisms from protections keys permits mechanisms to be examined by many reviewers without concern that the review itself compromise the safeguards.

- In cryptography, this is known as **Kerckhoff's principle**: a cryptosystem should be secure even if all aspects of the system except the keys being used are public knowledge.
- The open design principle is the opposite of the approach known as **security by obscurity**, which tries to achieve security by keeping cryptographic algorithms secret and which has been historically used without success by several organizations. Note that while it is **easy to change a compromised cryptographic key or password**, it is usually **infeasible to modify a system whose security has been threatened by a leak of its design**.

5. **Separation of Privilege (Duties)**

> A system should not grant permission based on a single condition.

A good example of this is multi-factor user authentication, which requires multiple techniques, say, password and smart card to authorize a user.

6. **Least privilege**

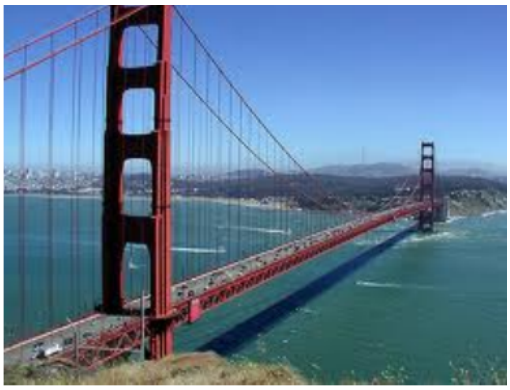
> Every process or user of the system should operate using the least set of privileges necessary to perform the task.

If this principle is enforced, abuse of privileges is restricted, and **the damage caused by the compromise** of a particular application or user account is **minimized**. For example:

- Most office works do not need the privileges of installing new software on a corporate computer, creating new accounts or sniffing network traffic. These employees can do their jobs with fewer privileges, e.g., access to office applications and a directory to store data.
- A web server's processes need not run with administrative privileges. They should run with the privileges of a less privileged user account.

7. **Least Common Mechanism (Isolation)**

> In systems with multiple users, mechanisms allowing resources to be shared by more than one user should be minimized.

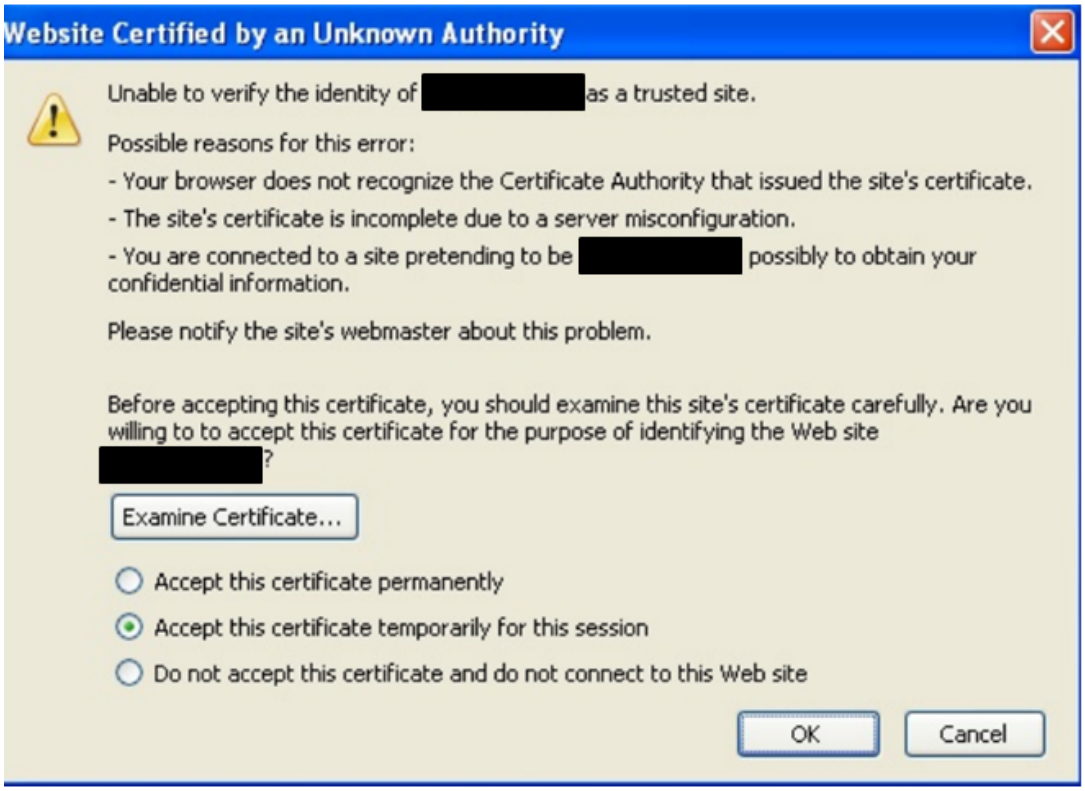


If a file or application needs to be accessed by more than one user, then these users should have separate channels by which to access these resources, to prevent unforeseen consequences that could cause security problems. The more state shared between different users, the more likely it is that this shared state may become a conduit for inadvertent (or malicious) information flow. For example:

- System surfaces touched by many users can become an attractive attack target for the adversarial user. In particular, if **one program can corrupt the shared state, it can then corrupt other programs which depend on it** and ultimately break security of the system.
- Amazon website: Denial of service attacks!!
- A covert channel is a type of computer security attack that creates a capability to **transfer information objects between processes that are not supposed to be allowed to communicate** by the computer security policy. For example, a covert storage channel can be established as follows: one process writes to a shared resource while another process reads from it.

8. **Psychological Acceptability (Usability)**

> Security mechanism should be easy to use.



The more difficult a security mechanism is to use, the more likely it is that users will circumvent it to get their job done or will apply it incorrectly, thereby introducing new vulnerabilities.

Most end users do not understand cryptographic mechanisms. They do not understand what a certificate is, its intended use and how to verify the authenticity of a server certificate. As a consequence, it is possible to impersonate a web server even in settings where server certificates are used.

Other Principles

1. **Defense in Depth** : Build redundant security mechanisms whenever feasible. Examples include multi-factor authentication, and DMZ.
2. **Minimum Trust** : Minimize trust. The difference between a trustworthy and a trusted system is important. If the users trusts the system, he assumes the system will satisfy this expectation. This is just assumption, however, and a trusted system may misbehave.

   In general, trust should avoided whenever possible. There is no guarantee that the assumptions made are justified. For example, in case of a system relying on external input, it should not trust that it is given only valid inputs; instead the system should verify that its input is indeed valid and take appropriate action when it's not the case. As another example, Lockheed-Martin was relying not only on the security of RSA's product (SecurID), but also on the security of RSA's own IT infrastructure and its ability to protect the originally issued seed values. Its trust in the security of RSA's own IT infrastructure was evidently misplaced; it was hacked.

3. **Work factor** : The cost of circumventing a security mechanism should be compared with the resources of an attacker when designing a security scheme.

   For example:

   - A system developed to protect student grades in a university database (potentially attacked by snoopers or students trying to change their grades) probably needs less sophisticated security measures than a system built to protect military secretes (potentially attacked by government intelligence organizations).

4. **Compromise recording** : It is sometimes more desirable to record the details of an intrusion than to adopt more sophisticated measures to prevent it.

   For example:

   - Internet-connected surveillance cameras are a typical example of an effective compromise record system that can be deployed to protect a building in lieu of reinforcing doors and windows.
   - The servers in an office network may maintain logs for all accesses to files, all emails sent and received, and all web browsing sessions.

This principle may not hold strongly on computer systems; it may be difficult to detect intrusion and attackers may be able to remove log entries.

References

1. [SS75] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. Proceedings of the IEEE, 63:1278–1308, 1975. pdf

Quiz Candidates for Security Overview

Drag the mouse for answers or use the following buttons: show answers hide answers

1. Fill one of security design principles (given by Saltzer and Schroeder) in the blank to match the descriptions:
   - _____ : Unless an entity is given explicit access to an object, it should be denied access to that object.
   - _____ : The security of a system should not depend on the obscurity of its protection mechanism.
   - _____ : A system should not grant permission based on a single condition.
   - _____ : Minimize the functions common to more than one user and depended on by all users.
   - _____ : Keep it simple.
   - _____ : Every access must be monitored and controlled.
   - _____ : Every process or user of the system should operate using the least set of privileges necessary to perform the task.
   - _____ : Security mechanism should be easy to use.

2. Consider the following general code for allowing access to a procedure:

```
int ret = IsAccessAllowd(...);

if (ret == ERROR_ACCESS_DENIED)
{
  // security check failed.
  // Inform user that access is denied.

}
else
{
  // security check is OK.
}
```

   a. Explain the security flaw in this program.

   b. What is the security design principle related to the above code?

   c. Rewrite the code to avoid the flaw
   Answer:

```
int ret = IsAccessAllowd(...);
if (ret == NO_ERROR)
{
  // Secure check OK.
  // Perform task.
}
else
{
```

```
    // Security check failed.
    // Inform user that access is denied.
}
```

3. Fill in the blank:

> The principle of Open Design by Saltzer and Schroeder is also known as Kerckhoff's principle : a cryptosystem should be secure even if all aspects of the system except the keys being used are public knowledge.

4. In a process called _____, a company allows nothing to run unless it is approved, whereas in a process called _____, the company allows everything to run unless it is not approved.

    a. whitelisting, encryption
    b. whitelisting, blacklisting
    c. encryption, whitelisting
    d. encryption, blacklisting
    e. blacklisting, whitelisting

    Answer:

5. Which security design principle is the following text referring to?

> One should be wary of performance improvement techniques that save the results of previous authorization checks, since permission can change over time. For example, an online banking web site should require users to sign on again after a certain amount of time, say 10 minutes, has elapsed. File systems vary in the way access checks are performed by application. It can be risky if permission are checked the first time a program requests access to a file, but subsequence accesses to the same file are not checked again while the application is still running.

    Answer:

6. Which security design principle is the following text referring to?

> The military concept of need-to-know information is an example of this principle. When this principle is ignored, then extra damage is possible from security breaches. For example, malicious code injected by the attacker into a web server application running with full administrator previleges can do substantial damage to the system.

    Answer:

7. Which security design principle is the following text referring to?

> Shared mechanisms may include cross-talk paths that permit a breach of data security, and it is difficult to make a single mechanism operate in a correct and trusted manner to the satisfaction of a wide range of users.

    Answer: