L12: Design Principles

Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgement

- Many slides are from or are revised from the slides of the author of the textbook
  - Matt Bishop, Introduction to Computer Security, Addison-Wesley Professional, October, 2004, ISBN-13: 978-0-321-24774-5. Introduction to Computer Security @ VSU's Safari Book Online subscription
  - http://nob.cs.ucdavis.edu/book/book-intro/slides/

Outline

- Overview
- Principles
  - Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Psychological Acceptability
- Case Analysis

Overview

- Principles for *the design and implementation of security mechanisms*
- Simplicity
  - Easy to understand designs and mechanisms
  - Less to go wrong with simple designs
  - Fewer possible inconsistencies within a policy or set of policies
- Restriction
  - Minimize access
  - Inhibit communication

Least of Privilege

- The *principle of least privilege* states that a subject should given only those privileges that it needs in order to complete its tasks
  - Function, not identity, controls the assignment of rights
  - Rights added as needed, discarded after use
  - Minimal protection domain for processes

Least of Privilege: Example

- In page 201, textbook
  - The UNIX operating system does not apply access controls to the user *root*. That user can terminate any process and read, write, or delete any file. Thus, users who create backups can also delete files. The *administrator* account on Windows has the same powers.
- Linux capabilities
  - See capabilities(7)

Least of Privilege: Example

- In page 202, textbook:
  - A mail server accepts mail from the Internet and copies the messages into a spool directory; a local server will complete delivery. The mail server needs the rights to access the appropriate network port, to create files in the spool directory, and to alter those files (so it can copy the message into the file, rewrite the delivery address if needed, and add the appropriate "Received" lines). It should surrender the right to access the file as soon as it has finished writing the file into the spool directory, because it does not need to access that file again. The server should not be able to access any user's files, or any files other than its own configuration files.

Fail-Safe Defaults

- The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object
  - Default access to an object is *none*
  - If action fails, system as secure as when action began
    - Undo changes the subject made in the security state of the system before it terminates.

Fail-Safe Defaults: Example

- In page 202, textbook
  - If the mail server is unable to create a file in the spool directory, it should close the network connection, issue an error message, and stop. It should not try to store the message elsewhere or to expand its privileges to save the message in another location, because an attacker could use that ability to overwrite other files or fill up other disks (a denial of service attack). The protections on the mail spool directory itself should allow create and write access only to the mail server and read and delete access only to the local server. No other user should have access to the directory. (to be continued)

Fail-Safe Defaults: Example (Continued)

- In page 202, textbook
  - In practice, most systems will allow an administrator access to the mail spool directory. By the principle of least privilege, that administrator should be able to access only the subjects and objects involved in mail queueing and delivery. As we have seen, this constraint minimizes the threats if that administrator's account is compromised. The mail system can be damaged or destroyed, but nothing else can be.

Economy of Mechanism

- The *principle of economy of mechanism* states that security mechanism should be as simple as possible
  - Keep it as simple as possible
    - The *KISS* Principle
  - Simpler means few possibilities exist for errors
    - And when errors occur, they are easier to understand and fix
  - Interfaces and interactions are particular suspect
    - Modules often make implicit assumptions about input or output parameters or the current system state

Economy of Mechanism: Example

- In page 203, textbook
  - The ident protocol sends the user name associated with a process that has a TCP connection to a remote host. A mechanism on host A that allows access based on the results of an ident protocol result makes the assumption that the originating host is trustworthy. If host B decides to attack host A, it can connect and then send any identity it chooses in response to the ident request. This is an example of a mechanism making an incorrect assumption about the environment (specifically, that host B can be trusted).

Economy of Mechanism: Example

- In page 203, textbook
  - The finger protocol transmits information about a user or system. Many client implementations assume that the server's response is well-formed. However, if an attacker were to create a server that generated an infinite stream of characters, and a finger client were to connect to it, the client would print all the characters. As a result, log files and disks could be filled up, resulting in a denial of service attack on the querying host. This is an example of incorrect assumptions about the input to the client.

Complete Mediation

- The *principle of complete mediation* requires that all accesses to objects be checked to ensure that they are allowed
  - *Should* check every access
  - Many systems usually done once, on first action
    - UNIX: access checked on open, not checked thereafter
    - If permissions change after, may get unauthorized access

Complete Mediation: Example

- In page 204, textbook
  - When a UNIX process tries to read a file, the operating system determines if the process is allowed to read the file. If so, the process receives a file descriptor encoding the allowed access. Whenever the process wants to read the file, it presents the file descriptor to the kernel. The kernel then allows the access. (to be continued)
  - If the owner of the file disallows the process permission to read the file after the file descriptor is issued, the kernel still allows access. This scheme violates the principle of complete mediation, because the second access is not checked. The cached value is used, resulting in the denial of access being ineffective.

Complete Mediation: Example (Continued)

- In page 204, textbook
    - If the owner of the file disallows the process permission to read the file after the file descriptor is issued, the kernel still allows access. This scheme violates the principle of complete mediation, because the second access is not checked. The cached value is used, resulting in the denial of access being ineffective.

Complete Mediation: Example

- In page 204, textbook
  - The Domain Name Service (DNS) caches information mapping host names into IP addresses. If an attacker is able to "poison" the cache by implanting records associating a bogus IP address with a name, one host will route connections to another host incorrectly. Section 13.6.1.2 discusses this in more detail.

Open Design

- The *principle of open design* states that the security of a mechanism should not depend on the secrecy of its design or implementation
  - No to “Security through obscurity”
  - Does not apply to information such as passwords or cryptographic keys because passwords and keys are not *algorithms*
  - Popularly misunderstood to mean that source code should be public

Open Design: Example

- In page 205, textbook
  - The Content Scrambling System (CSS) is a cryptographic algorithm that protects DVD movie disks from unauthorized copying. The DVD disk has an authentication key, a disk key, and a title key. The title key is enciphered with the disk key. A block on the DVD contains several copies of the disk key, each enciphered by a different player key, and a checksum of the disk key. When a DVD is inserted into a DVD player, the algorithm reads the authentication key. It then deciphers the disk keys using the DVD player's unique key. When it finds a deciphered key with the correct hash, it uses that key to decipher the title key, and it uses the title key to decipher the movie. The authentication and disk keys are not located in the file containing the movie, so if one copies the file, one still needs the DVD disk in the DVD player to be able to play the movie. (to be continued)

Open Design: Example (Continued)

- In page 205, textbook
  - In 1999, a group in Norway acquired a (software) DVD playing program that had an unenciphered key. They also derived an algorithm completely compatible with the CSS algorithm from the software. This enabled them to decipher any DVD movie file. Software that could perform these functions rapidly became available throughout the Internet, much to the discomfort of the DVD Copyright Control Association, which promptly sued to prevent the code from being made public. As if to emphasize the problems of providing security by concealing algorithms, the plaintiff's lawyers filed a declaration containing the source code of an implementation of the CSS algorithm. When they realized this, they requested that the declaration be sealed from public view. By then, the declaration had been posted on several Internet sites …

Separation of Privilege

- The *principle of separation of privilege* states that a system should not grant permission based on a single condition
    - Require multiple conditions to grant privilege
        - Separation of duty
        - Defense in depth

Separation of Privilege: Example

- In page 206, textbook
  - On Berkeley-based versions of the UNIX operating system, users are not allowed to change from their accounts to the root account unless two conditions are met. The first condition is that the user knows the root password. The second condition is that the user is in the wheel group (the group with GID 0). Meeting either condition is not sufficient to acquire root access; meeting both conditions is required.
  - How about Linux?

Least Common Mechanism

- The *principle of least common mechanism* states that mechanisms used to access resources should not be shared
  - Mechanisms should not be shared
    - Information can flow along shared channels
    - Covert channels
  - Isolation
    - Virtual machines
    - Sandboxes

Least Common Mechanism: Example

- In page 206, textbook
  - A Web site provides electronic commerce services for a major company. Attackers want to deprive the company of the revenue it obtains from that Web site. They flood the site with messages and tie up the electronic commerce services. Legitimate customers are unable to access the Web site and, as a result, take their business elsewhere. (to be continued)
- Here, the sharing of the Internet with the attackers' sites caused the attack to succeed. The appropriate countermeasure would be to restrict the attackers' access to the segment of the Internet connected to the Web site. Techniques for doing this include proxy

Least Common Mechanism: Example (Continued)

- In page 206, textbook
  - Here, the sharing of the Internet with the attackers' sites caused the attack to succeed. The appropriate countermeasure would be to restrict the attackers' access to the segment of the Internet connected to the Web site. Techniques for doing this include proxy servers such as the Purdue SYN intermediary or traffic throttling. The former targets suspect connections; the latter reduces the load on the relevant segment of the network indiscriminately.

Psychological Acceptability

- The *principle of psychological acceptability* states that security mechanisms should not make the resource more difficult to access than if the security mechanisms were not present
  - Security mechanisms should not add to difficulty of accessing resource
    - Hide complexity introduced by security mechanisms
    - Ease of installation, configuration, use
    - Human factors critical here

Psychological Acceptability: Example

- In page 207, textbook
  - The *ssh* program allows a user to set up a public key mechanism for enciphering communications between systems. The installation and configuration mechanisms for the UNIX version allow one to arrange that the public key be stored locally without any password protection. In this case, one need not supply a password to connect to the remote system, but will still obtain the enciphered connection. This mechanism satisfies the principle of psychological acceptability.

Psychological Acceptability: Example

- In page 207, textbook
  - When a user supplies the wrong password during login, the system should reject the attempt with a message stating that the login failed. If it were to say that the password was incorrect, the user would know that the account name was legitimate. If the "user" were really an unauthorized attacker, she would then know the name of an account for which she could try to guess a password.

Psychological Acceptability: Example

- In page 207, textbook
  - A mainframe system allows users to place passwords on files. Accessing the files requires that the program supply the password. Although this mechanism violates the principle as stated, it is considered sufficiently minimal to be acceptable. On an interactive system, where the pattern of file accesses is more frequent and more transient, this requirement would be too great a burden to be acceptable.

Psychological Acceptability: Example

- Security Enhanced Linux?
- Access databases in a 3-tier web application?

Key Points

- Principles of secure design underlie all security-related mechanisms
- Require:
  - Good understanding of goal of mechanism and environment in which it is to be used
  - Careful analysis and design
  - Careful implementation

Summary

- Principles
  - Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Psychological Acceptability