Security Design Concepts

Target Course

Software Engineering, Software Design

Learning Goals

A student shall be able to:

1. Describe security design principles and identify security issues associated with common threats and attacks.
2. Apply principles of secure design and defensive programming techniques when developing software.

IAS Outcomes

The CS2013 Information Assurance and Security outcomes addressed by this module are:

IAS Knowledge Topic	Outcome
Principles of Secure Design	1. Describe the principle of least privilege and isolation as applied to system design. [Familiarity] 2. Summarize the principle of fail-safe and deny-by-default. [Familiarity] 3. Discuss the implications of relying on open design or the secrecy of design for security. [Familiarity] 4. Explain the goals of end-to-end data security. [Familiarity] 5. Discuss the benefits of having multiple layers of defenses. [Familiarity] 7. Describe the cost and tradeoffs associated with designing security into a product. [Familiarity] 8. Describe the concept of mediation and the principle of complete mediation. [Familiarity] 9. Describe standard components for security operations, and explain the benefits of their use instead of reinventing fundamentals operations. [Familiarity] 11. Discuss the importance of usability in security mechanism design. [Familiarity]

Dependencies

- A student understands the cybersecurity topics covered in Input Validation and Principles.

Summary

Discuss security design principles as it relates to the design and development of software applications and systems.

Estimated Time

40 minutes.

Materials

What are the security principles that should be adhered to when designing and implementing a system?

The first ten principles come from a paper written by Saltzer and Schroeder in 1975 [1].

1. Economy of mechanism — Keep your design as simple as possible (aka KISS-Keep It Simple Stupid). This allows quality assurance methods (e.g., formal reviews, design walkthroughs) to have the greatest chance of finding security vulnerabilities.
2. Fail-safe defaults — The default setting/action should be to favor security over usability. When in doubt, deny access. That is, your design

	should base access to data on permission rather than exclusion. Only when the protection scheme identifies conditions to permit access should the data be accessible. In contrast, creating a scheme that describes the conditions for refusing access presents the wrong psychological perspective for a secure software design. To state another way, the rules needed to express permission are likely to be simpler to understand than the rules needed to refuse permission.
3. Complete mediation	Every access request should be checked for adherence to a protection scheme. This implies that a foolproof method of identifying the source of each request must be devised and suggests that ideas about improving performance by remembering the result of a previous authority check be examined skeptically. Care must be taken when a change in authority occurs, to ensure that the remembered results are systematically updated.
4. Open design	Publish your design for anyone to review. This allows reviewers to comment on the security mechanisms being used while protecting the keys or passwords that are used by the mechanisms. You should assume that your design is not a secret.
5. Separation of privilege	Originally defined as requiring multiple conditions to access a restricted resource or to perform some action (e.g., require two or more keys to unlock a protection mechanism). More recently, this has been defined as separating components of a system to reduce damage when a security breach occurs in any one component.
6. Least privilege	Each user and program should operate with the minimum set of privileges necessary to accomplish the job. When software must access an information asset, it should ideally be granted this access only for the moments in time when it is using the information asset. This limits the damage that may result from an accident or error.
7. Least common mechanism	Minimize the number of resources being shared/used by more than one user or system. Each security mechanism that is shared among most/all users or is shared among systems, especially when shared variables are used, represents a potential information path that may unintentionally compromise security. Do not share objects and protection mechanisms, instead create separate instances for each user or system interface.
8. Psychological acceptability	User interfaces related to security mechanisms should be designed based on what a user expects. A well designed HCI will match the protection mechanism to the user's mental image of their protection goals.

9. Work factor	The cost of compromising a security mechanism should be compared with the resources of an attacker when designing a security scheme. When a likely attacker has limited resources, the system may require less sophisticated defensive mechanisms.
10. Compromise recording	It may be more desirable to record the details of an intrusion rather than designing more sophisticated prevention mechanisms.

In 2013, Gary McGraw [2] expanded on the security principles identified by Saltzer and Schroeder by adding five more principles[1].

11. Secure the weakest link	The suite of security mechanisms being used are only as good as the weakest security mechanism being used. The analogy often used is that a chain is only as strong as its weakest link. Likewise, a system is only as secure as its weakest security mechanism.
12. Defend in depth	Your design should include redundancy and layers of defense. This approach looks to manage security risks by using a diverse set of security mechanisms that provide redundant capabilities or are provided by different software layers.
13. Be reluctant to trust	Be skeptical of security protections that are not within your software system. The quote "trust but verify" often used to describe international agreements to limit nuclear weapons is a good motto to follow when it comes to placing your software within an operating environment. A cloud provider may claim to provide certain protections, but it is best to verify these as best you can.
14. Promote privacy	Your design needs to consider the types of personal information you are collecting from a user. Do you really need the information you are requesting? Should the personal information be encrypted? Does this data really need to be persistently stored?
15. Use your resources	Nobody knows everything about what a good software security design looks like. Talk to others about the design choices you are making. Have experts with different backgrounds review your design.

What are the common access control models?

• Access control matrices	A table that defines permissions for each resource. Each row identifies the users and systems needing access to resources while each column identifies each resource (e.g.,

[1] McGraw includes the first 8 principles from Saltzer and Schroeder then adds 5 principles to get to 13. Thus, McGraw's article is titled "Thirteen Principles …".

	file, directory, device). Each cell shows the access rights (e.g., read, write, execute) for the user/system specified at the start of the row and the resource identified at the top of the column.
• Access control lists	A list of users and systems that have access rights to a resource. Each resource has a separate list.
• Capabilities	A list of resources and access rights. Each user or system has a separate list.
• Role-based access control	Roles are defined and given access rights. Users and systems are then assigned one or more roles.

What are the cryptographic concepts that should be understood and used?

• Cryptosystems	Encryption and decryption of data. Symmetric encryption uses the same secret key to encrypt and decrypt the data. Asymmetric encryption (aka public-key encryption) uses a public key to encrypt the data and a private key to decrypt the data.
• Digital signatures	Use public-key encryption to verify who sent you the data. The sender encrypts the data using their private key. Anyone receiving this data may use the sender's public key to decrypt the data. In theory, only the sender knows their private key and so this verifies that the data did come from the sender.
• Simple attacks on cryptosystems	See the module ***Common Attack Types***.
• Cryptographic hash functions	A type of checksum on a data value that has two important properties: the hash function is a *one-way* function and the checksum generally contains many fewer bits than the original data value. A one-way hash function produces a checksum given a data value. But it is hard to recreate the data value if all you have is a checksum value.
• Digital certificates	A statement from a certificate authority that combines a public key with identifying information about the entity that owns that public key. This is used to ensure that the public key being used is associated with the entity you want to communicate with.

What issues exist in correctly implementing and using computer security mechanisms?

• Efficiency and usability	Providing security mechanisms that are slow gives a user a disincentive to use these mechanisms. Using a security mechanism that is easily misunderstood by a user allows for greater potential in misuse, which may result in vulnerabilities.
• Passwords	A primary authentication mechanism. Ideally, passwords should be hard to guess but easy to remember.

• Social engineering	A collection of attacks that take advantage of the trust people place in information systems. Examples include phishing, spear phishing, spoofing, pretexting, and quid pro quo. These types of attacks are the digital equivalent of traditional methods (i.e., con, scam,) to defraud a person/group after gaining their trust.
• Vulnerabilities from programming errors	A design should provide clear instructions on how to implement security mechanisms and how to test a system against all security requirements.

Assessment Strategies

Short answer questions that have been included in a take-home final exam for a Software Design Course are listed below. ***A possible answer is included in bold italics.***

In a Word document, answer the following questions. Please identify each of your answers using the outline numbering for each question.

1. In a few sentences, explain the security design principle of *work factor*.

   The security design principle of work factor says that the cost of developing security controls/mechanisms should be balanced with the cost an adversary/threat would incur finding a way into your system. When an adversary would need to spend very little time/money to find a way into your system, perhaps it is better to simply implement compromise recording (i.e., log events).

2. In a few sentences, explain the security design principle of *least common mechanism*.

   The security design principle of least common mechanism says that different subsystems/components in your software design should utilize different security controls/mechanisms. An adversary/threat may then need to penetrate each of these controls/mechanisms before they gain unauthorized access to your system/data.

3. For each of the following, explain how the security design principles of *compromise recording* and *least privilege* may or may not apply.
   a) Use of an ATM (automated teller machine).

      Compromise recording: each of your transactions will be recorded/logged. This would include each inquiry, deposit, and withdrawal as well as recording each login event. Each log entry will identify you (likely by account number).

      Least privilege: upon successful login, you would have access to your accounts. However, you would not have access to accounts that you do not own. If you cannot successfully login, you have access to no accounts.

   b) Purchasing an item at a retail store with cash while using a customer loyalty card.

      Compromise recording: your transaction will be recorded/logged. This would include the merchandise purchased and the amount, along with your identifying information (e.g., loyalty card account number, name).

Least privilege: this scenario has no impact on least privilege as there is no authorization needed when using a customer loyalty card. A person could use anyone's loyalty card.

4. How would a formal review (aka: inspection) of a software design artifact impact the software security design principle *economy of mechanism*?

   A formal review (inspection) impacts economy of mechanism by allowing inspectors to identify complexity that should be reduced or removed from the design.

5. What are the benefits and limitations of the security design principle *open design*?
   a) Benefits:

      The benefits of open design are that anyone may comment on your design. This allows experts from outside your organization to find vulnerabilities and defects that may be in your design.

   b) Limitations:

      The limitations of open design include an inability to make the design open due to a competitive advantage policy that forbids designs from being publicized. Also, you have no control over who reviews your design. So the designs could be viewed by individuals that do not have the expertise to comment on its security implications.

6. When thinking about the security design principles *defend in depth* and *be reluctant to trust*:
   1. What do these have in common?

      When using third-party software within your solution, you should understand and test the security features in this software. In addition, other parts of your design should include redundant mechanisms in case the third-party software contains security mechanisms that are too weak, not implemented correctly, or are changed over time.

   2. How are these different?

      Defend in depth says that your design should have redundant security features at different layers of the software. In contrast, be reluctant to trust talks about third party software and a willingness to verify the capabilities of this software.

Multiple choice and true/false, and short answer questions that have been included in quizzes and final exams for a Networks Course, with an emphasis on secure software development, are listed below.

1. Why is the application layer so important with regarding to meeting security goals and applying security concepts?
   a. Because the other layers satisfy many but not all of the goals/concepts.
   b. Because the other layers were developed by someone else, and we should never rely on third-party software to satisfy our security goals/concepts.
   c. All of the above.

d. None of the above.

ANSWER: d

2. Which of the following is the best description for the security design principle *complete mediation*?
   a. Each request to access data should be checked for adherence to a protection scheme.
   b. Each request to access data should be checked for adherence to an authentication scheme.
   c. Each request to access data should be checked for adherence to an authorization scheme.
   d. None of the above.

ANSWER: a

3. The security design principle of psychological acceptability means that the user interface for a security mechanism conforms to the user's expectations.
   a. True.
   b. False.

ANSWER: a

4. The security design principle of complete mediation means that each request to access data/system should be checked for adherence to a protection scheme. This design principle should be designed into all systems, regardless of the impact to performance or usability.
   a. True.
   b. False.

ANSWER: b

5. Proprietary designs are inherently more secure than open designs since fewer people know about the proprietary designs.
   a. True.
   b. False.

ANSWER: b.

6. HTTPS uses public key cryptography to establish a secret key between the client and server.
   e. True.
   f. False.

ANSWER: a

Short answer questions that have been included in quizzes and final exams for a Networks Course, with an emphasis on secure software development, are listed below. ***A possible answer is included in bold italics.***

7. Identify two security risks/issues if a distributed software application does not support the security goal of confidentiality?

If the security goal of confidentiality is not considered when developing a distributed application, then an individual may be able to:

- ***Gain access to persistent data that is in plaintext form (i.e., has not been encrypted).***
- ***Gain access to data in transit that is in plaintext form (i.e., has not been encrypted).***
- ***Gain access to data that they are not authorized to view, update, or delete.***

This may lead to:

- ***Trust in the organization's systems being decreased.***
- ***User's stop using the app since the app is less safe; their trust is lowered.***
- ***A malicious actor stealing one's identify.***
- ***A malicious actor leaking personal data to the public.***

8. What is the impact to the security of a distributed software application if its design does not consider the security goal of integrity?

If the security goal of integrity is not considered when developing a distributed application, then an individual may be able to:

- ***Create, update, or delete data even though they do not have the authority to do so.***
- ***Create a fake file (i.e., spoofing) that includes malware. A user downloading this file may behn get infected with the malware.***

This may lead to:

- ***A lack of trust in the distributed application.***

9. Identify a reason why the security goal of availability is so hard to achieve from the perspective of developing a distributed software application?

The security goal of availability is so hard to achieve when developing a distributed application since:

- ***A DDOS (distributed denial-of-service) attack is able to overwhelm the application with packets that are not valid, preventing the application from receiving packets that contain legitimate data. The design of the distributed software application cannot do anything to prevent the invalid packets from being received; other network protocol layers and devices must deal with this type of attack.***
- ***The other goals may be in direct conflict with availability. For example, two-factor authentication will increase confidentiality by requiring a second method for proving who you are, but this second factor may not be working. This prevents***

user from accessing the application, resulting in the application being more secure but less available.

10. Explain why it is hard to develop a distributed software application that supports both non-repudiation and anonymity?

Developing a distributed software application that supports non-repudiation means that user actions would be recorded in a way that could be reviewed by others, which would prevent a user from performing application actions that are intended to be anonymous.

11. Assurance is a belief an individual has that a system is trustworthy. This trust is managed by the policies, permissions, and protections the system implements.
    a. Give an example of a policy, along with its appropriate permission(s) and/or protection(s), that will provide assurance to a user of a system.
    b. Describe a risk that may affect the policy you described in 14.a, and identify how this risk affects the permission(s) and/or protection(s) associated with the policy.

- ***A network policy that all data transmitted within an organization's internal network be encrypted using symmetric cryptography. An associated permission would be who (i.e., people and systems) has access to the symmetric key used to encrypt and decrypt the data. An associated protection would be how the entities that have permission to access to the symmetric key get authenticated to ensure they are who they say they are.***
  - ***A risk of this policy is when the symmetric key is compromised, allowing anyone with the key to encrypt and decrypt data being transmitted on the internal network. This risk may be realized through a weak authentication system that is used to protect access and use of the key.***
- ***Using HTTPS with a login screen. The user can only access their data after they login. If they try to use HTTP deny access.***
  - ***If the certificate authority (for the SSL cert.) was compromised, the website may be easier to spoof, along with the certificate. Protection: alert the user to a breach.***
- ***A company with a policy to require two-factor authentication and a promise to never disclose personal info to ad sites can assure users that their info on this company's systems is secure.***
  - ***If a user gets their password broken or stolen, this breaks the policy of keeping user data confidential, because the malicious actor now has permission to access the account info. They can, if things aren't implemented correctly, reset or remove two-factor authentication and other protections on the account.***

12. As a designer of software applications, what two questions that you should asking and discussing to help promote privacy of data?

- ***General comments:***
  - ***These questions should align with one or more of the security goals and concepts OR should be related to one or more of the 15 security design principles.***
- ***Some examples:***

- ***Do we really need to obtain and either store or transmit this private data?***
- ***Which of this data may be used maliciously to identify our users/customers?***
- ***Which of this data should we encrypt while it is persistently stored?***
- ***Which of this data should we encrypt while it is in transit?***
- ***What type of cryptographic algorithm should we be used to encrypt/decrypt the data?***

References

[1] J.H. Saltzer & M.D. Schroeder, (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), 1278-1308.

[2] G. McGraw, (2013). Thirteen principles to ensure enterprise system security. Retrieved on July 28, 2015 from searchsecurity.techtarget.com/opinion/Thirteen-principles-to-ensure-enterprise-system-security.