Security Module: Security Design Patterns

Description

Software design patterns are solutions to problems that arise regularly during software design. They are meant to serve as readily applicable, time-saving strategies for software development. The structured documentation that accompanies a properly defined pattern allows developers to quickly identify and apply patterns to a given problem. Security patterns are software design patterns that describe security mechanisms such as logging and access control. These patterns can be applied to solve numerous security concerns.

Objective

The objectives of this security module are to describe a few security design patterns and illustrate situations that the patterns could be used.

Activities

There are two activities for this module. The first is a discussion assignment on security and security design patterns, software design patterns that describe security mechanisms such as logging and access control. Second, students are given three scenarios to evaluate, they must pick a security design pattern discussed in the first assignment that best addresses security concerns detailed in the scenario.

Module Contents

- Assignment 1: Security Design Pattern Discussion
- Assignment 2: Applying Security Design Pattern
- Security Hierarchy
Assignment 1: Security Design Pattern Discussion

Security

Security in software systems is the protection of an enterprise’s data against unwanted access or modification. Security is a diverse challenge that includes practices that range from escaping user input to protecting against unauthorized access to sensitive information.

Security is traditionally characterized as a non-functional requirement (NFR), a requirement for how a system should operate. Security, unfortunately, is often engineered into systems late in the design process or after implementation. This results in a system that is less maintainable and less trustworthy in terms of putting an enterprise’s data at risk. To better understand security, it can be decomposed into a hierarchy of more precise NFRs. The first decomposition results in four NFRs: confidentiality, integrity, availability, and accountability.

Requirements for the confidentiality of an enterprise’s data are typically derived from that enterprise’s security policies as they relate to this data. These requirements, as noted in, can be further decomposed into requirements for user authentication and authorization. Requirements for confidentiality should strongly influence how a system is designed, how access is controlled, how processes are controlled, and how users are authenticated. Breaches of confidentiality can often be attributed to mistakes in system implementation that are exploited by malicious parties to gain unauthorized access to the compromised data. An example of a recent breach in confidentiality occurred in 2011 when hackers infiltrated Sony’s PlayStation Network resulting in the theft of over 77 million account holders’ personal information. This breach has cost Sony over $171 million, in terms of lost revenues and costs to secure its networks.

Confidentiality is the protection of an enterprise’s sensitive data from unauthorized access. Authentication is the confirming of a user’s or entity’s identity. Authorization can be done through physical confirmation, using devices such as biometrics or key card access, or specialized information like a secure password or a unique authentication key. Authorization is the granting of access to system resources such as accounts, files, or processes. Authorization is typically granted after a user has been authenticated. What resources a user is authorized to use may be stored in an access control list or based on a user’s role within the system.

Requirements for the integrity of an enterprise’s data are derived from that enterprise’s operating policies. Chung et al. cite data’s completeness, precision, and validity as considerations in assessing its integrity. A fourth concern is timeliness, or the extent to which a system’s information is current. Policies for data integrity must clearly define who can modify what data. Data corruption can be attributed in part to the deliberate modifications to datasets or processes by unauthorized users: for example, the alteration of experimental data to suggest a desired outcome. This corrupt data, when used by other processes or systems,
can further contaminate other datasets and processes.

Integrity is the protection of an enterprise’s data from unauthorized modification. Completeness is the storing of all information that a system is required to have. Precision is the degree to which stored information models true or accepted values. Timeliness is the currency of the system’s information. Validity is the correctness of that information, relative to the entities that it is intended to model.

Requirements for the availability of an enterprise’s services are derived from that enterprise’s quality service policies. Barbacci, for example, notes that “availability is required for security critical aspects of any given system.” These requirements can be further decomposed into requirements for usability, reliability, and compatibility. Resources can be rendered unusable by malicious or inadvertent denial of service attacks: bombardments of resources with more requests than they can handle.

Availability is the protection of an enterprise’s data from becoming unavailable from authorized users. Usability is the amount of time and effort required to use a program. Reliability is a system’s ability to function in adverse circumstances, including attacks and component failures. Compatibility is the degree to which a program operates securely in a given environment.

Requirements for the accountability of an enterprise’s services can be derived from an enterprise’s policies for employees’ responsible for their actions. Requirements for accountability can be decomposed into requirements for logging, monitoring, and reporting users’ actions. Logging allows the enterprise to determine which user may have breached system confidentiality or affected data integrity. For example, medical record systems will log medical providers who access patient information. This log can be used to determine if a provider breaches patient confidentiality by accessing records of individuals in ways that are not allowed by government regulations.

Accountability is the association of actions that affect the enterprise’s data with the person or persons who perform those actions. Logging is the recording of actions performed in or by a software system. Monitoring is a system’s awareness of actions that may require notification to a third party. Reporting is the relaying of information to a third party.

**Security Design Patterns**

Software design patterns are solutions to problems that arise regularly during software design. They are meant to serve as readily applicable, time-saving strategies for software development. The structured documentation that accompanies a properly defined pattern allows developers to quickly identify and apply patterns to a given problem. Security patterns are software design patterns that describe security mechanisms such as logging and access control. These patterns can be applied to solve numerous security concerns.
This decomposition can be used as a tool to categorize security design patterns. To illustrate this, a set of patterns is described below that can be categorized as authentication under confidentiality.

**Authorization**

The Authorization pattern identifies the subjects that may access a given resource, along with a subject’s access privileges for that resource. This pattern separates permissions from the subject and resources. It is flexible enough to handle a variety of subjects, resources, and privileges. Security policies should define privilege sets for subjects that will be encapsulated in authorization rules. These authorization rules will be associated with a subject and resource.

![Figure 1: Authorization Pattern](image)

Figure 1 illustrates the Authorization pattern with three participants: the Subject, Permission, and Resource. The Subject is the entity that needs to access the resource. Each Permission represents one way in which the Subject can access the Resource and provides mechanisms to check access.

**Multilevel Security**

The Multilevel Security pattern determines access permissions for data objects by partitioning users and data into categories, based on patterns of acceptable use. It uses trusted processes to change these classifications at need. Classifications for users are called clearances, while classifications for data are called sensitivity levels.

The use of classifications for users and data protects the system’s confidentiality and integrity. This pattern’s use should be limited to organizations where permissions are based on rank or position within an organization.
Figure 2: Multilevel Security Pattern

Figure 2 illustrates the Multilevel Security pattern with seven participants: Subject, Data, TrustedProcess, Category, Clearance, Category, and Classification. The Subject contains the Category for the organizational unit to which it belongs and the Clearance. Clearance represents the Subject’s clearance level; it is used to determine access permissions. The Data also contains the Category for the organizational unit to which it belongs and the Classification. Classification represents the Data’s sensitivity level; it is used to determine the access permissions. The TrustedProcess is the entity that can change the Clearance level for the Subject and Classification level for the Data.

**Role-Based Access Control**

The Role-Based Access Control pattern associates permissions with users based on their system-assigned roles. Associating user with roles and roles with permissions eliminates the work of associating individual users with individual sets of permissions. This pattern is appropriate when a large number of users or a large number of resources share related access privileges.
Figure 3: Role-Based Access Control Pattern

Figure 3 illustrates the Role-Based Access Control pattern with four participants: User, Role, PermissionsTo, and DataObject. User is the class that represents the system’s current user. Role represents the current role the user has assumed. DataObject represents the data the user wishes to access. An association class, PermissionsTo, determines the role’s permissions for the DataObject. Note that the User has a many-to-many relationship to the Role, and Role has a many-to-many relationship with DataObject.

This pattern reduces the workload for managing users and simpler security, since there are many more users than roles. This pattern also allows users to switch between roles, making the application easier to use.

Roles
The Roles pattern consolidates security policies that apply to multiple users into a single entity. Associating privileges with roles instead of individual users makes it easier to manage groups of users with common privileges. A strategy for implementing Roles is to allow a user to have multiple roles and a role to have multiple privileges. This implementation allows each role to represent a stakeholder and to capture the stakeholder’s appropriate privileges. Role-based inheritance relationships can also be used to form a hierarchy of roles.

One consequence of the Roles pattern is that administrators manage user-role and role-privilege relationships instead of user-privilege relationships. Associating multiple users to a role is simpler than associating users with individualized sets of privileges. Other consequences include the system having a good way to group privileges and it being more convenient to administer. These benefits come at the cost of adding an extra layer of complexity to the system. One well known application of Roles is UNIX’s owner-group-other role system. Roles are also known as Actors, Groups and Profiles.
Figure 4: Roles Pattern

Figure 4 illustrates the Roles pattern with three participants: User, Role, and Privilege. A User is associated with a Role. Once the user is associated with a Role, that user gains Privileges associated with the Role.

**Secure State Machine**

The Secure State Machine pattern provides a “clear separation between security mechanisms and user-level functionality by implementing the security and user-level functionality in two different state machines”. By separating security and user-level functionality, designers increase the design’s cohesion, making it easy to test, review, and verify security properties. This decreases the likelihood of introducing vulnerabilities into the system. The pattern also decreases coupling between the security and user-level functionality, making future modifications to the system easier.
Figure 5 illustrates the Secure State Machine pattern with four participants: Secure Context, Secure State, User Function Context, and User Function State. Secure Context, the primary participant, provides all operations to the client and acts as a proxy to the User Function Context. Secure State is an abstract class that must define all functionality handled by the secure state machine. Secure State will be subclassed by several concrete classes that represent how the security state changes based on the system’s current user. User Function Context is a class whose functionality must match the Secure State class so that requests can be forwarded to the User Function State when security requirements are met. User Function Context can only be created by Security Context. User Function State is an abstract class that must have the same interface as the Secure State. The Secure State Machine pattern “separate[s] security mechanics from user-level functionality” and “prevents programmatic access to the user-level functionality that avoids security” checks.
Questions

1. Draw the hierarchy for security discussed in this assignment.

2. Define security and give a recent security breach that made headlines.

3. List and define the NFRs that security is decomposed to in this assignment.

4. What security issue do the patterns detailed in this assignment address?

5. What is a design pattern?

6. What is a security design pattern?
Assignment 2: Applying Security Design Patterns

In this assignment, you will be given three scenarios where a system needs to be developed to solve a clients problem. The scenario’s problem description will be short and incomplete, but should have enough information to help you decide what pattern could be used. You are to choose one of the five patterns described above and defend your choice.

Scenario 1: US Department of Defense

The United States Department of Defense (DoD) needs a system to manage audio files that were obtained using wire taps. These audio files contain information that can be classified based on its sensitivity. These classifications include unclassified, confidential, secret, and top secret. To obtain access to a file with a top secret classification an individual will need to have top secret clearance. Individuals who have access to top secret audio files also have access to unclassified, confidential, and secret files. Individuals who have access to secret audio files also have access to unclassified and confidential files, but not top secret files. Individuals who have access to confidential audio files also have access to unclassified files, but not secret or top secret files. Unclassified audio files do not require a clearance level.

Questions

1. Which security design pattern do you think best fits the scenario?

(a) Authorization  (d) Roles
(b) MultiLevel Security
(c) Role-Based Access Control  (e) Secure State Machine

2. Defend you choice.
Scenario 2: ETSU Academic Affairs Office

The Academic Affairs Office at East Tennessee State University (ETSU) needs a system to handle the intent to graduate application forms. This application form has five sections that have to be completed by five different people. The first section contains information the student completes that has his/her enumeration, name, concentration, and semester/year of graduation. After that section has been completed it must be sent to the department chair who signs off that the student has met all department requirements. Next, the college dean signs off that the student has met all college requirements. Finally, the application has to be sent to the Bursar's office and the ETSU library to check for outstanding fees. Once the student has completed his/her section and all other sections have been completed the student may graduate.

The system needs to accommodate several hundred applications that must go through several dozen departments contained in eight colleges. The system will have hundreds of users interacting with the system for the various departments and colleges. The client has requested that adding and removing user from the system requires little effort. This means assigning permissions to individual users would require too much work.

Questions

1. Which security design pattern do you think best fits the scenario?

   (a) Authorization           (d) Roles
   (b) MultiLevel Security
   (c) Role-Based Access Control           (e) Secure State Machine

2. Defend your choice.
Scenario 3: Tennessee Bee Keepers

Tennessee Bee Keepers (TBK) is a company that rents beehives to farmers in Tennessee. These beehives are used to pollinate fields. This process results in a honey that the beehive produces. The owner of TBK needs a system to manage his hives. He wants to track each beehive to monitor the honey yield to optimize placement. Honey yields can vary based on the crop pollinated in the fields. TBK has employees who will interact with the system. These employees will input data into the system, but will not have access to reports or other sensitive information. TBK’s owner, Tom, is the only user that will have access to reports and sensitive information. Tom would also like to track employee information as well. Tom’s wife will manage employee information within the system.

The system needs to accommodate several hundred beehives that the user can access information on. The system will need to support three employees, the company’s owner, and the owner’s wife. Information about the beehives is not sensitive, but the employee information is.

Questions

1. Which security design pattern do you think best fits the scenario?

   (a) Authorization
   (b) MultiLevel Security
   (c) Role-Based Access Control
   (d) Roles
   (e) Secure State Machine

2. Defend your choice.