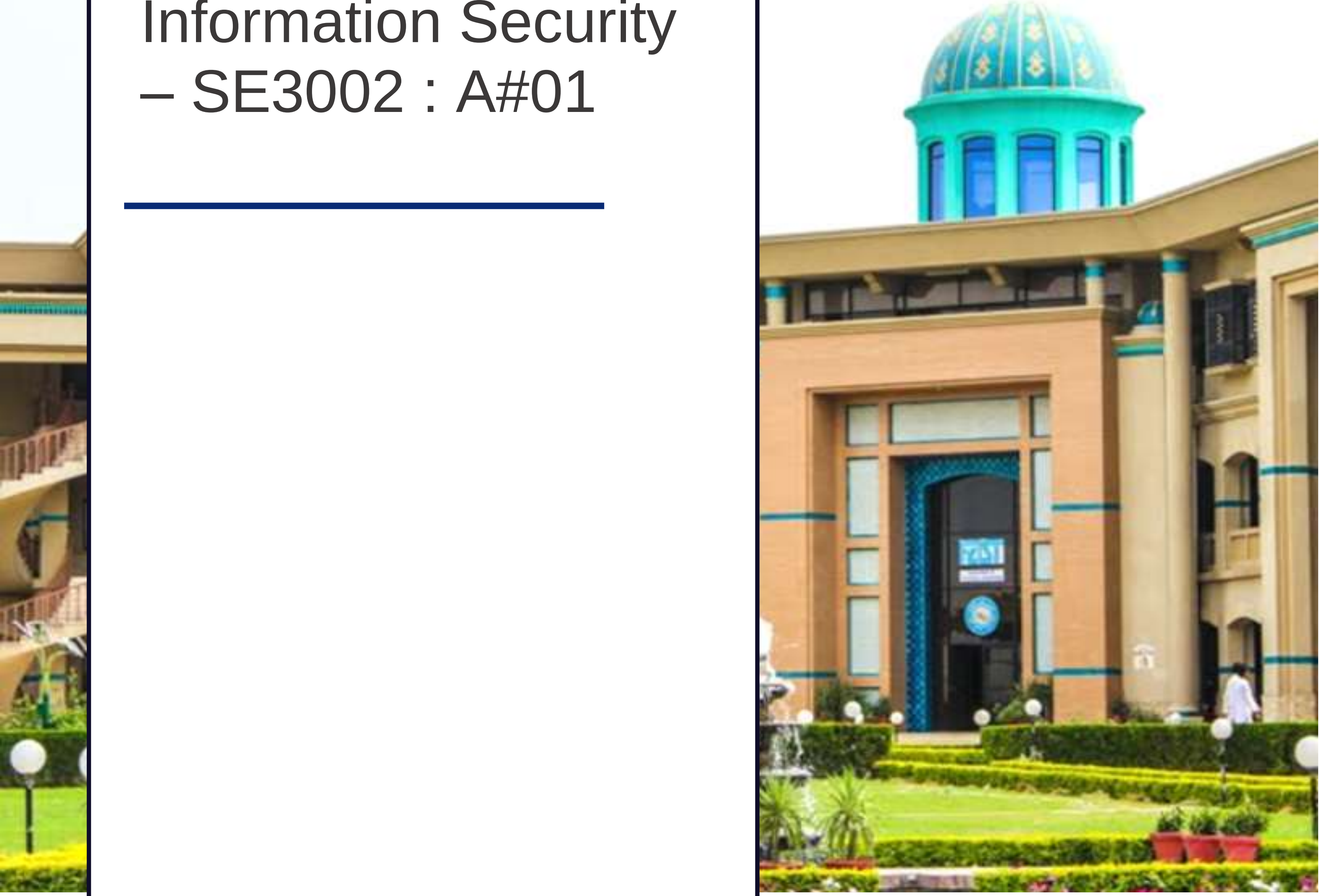

# Information Security – SE3002 : A#01

**FEBRUARY 17, 2023**

Daniyal Khan : 20i-1847
Mahad Rahat : 20i-1808

## Contents: Assignment 01

## Question:01

- Part 1

  - Following the Principle of least privilege

```
function getUser(id) {
  //fetching a user object in a DB
  const user = DB.findUser(id);
  if (user)
    return {
      name: user.name,
      email: user.email
    }

  return null;
}
```

In this code example, the getUser function retrieves user data from a database or API based on the id parameter. It then returns a new object containing only the user's name and email address. This approach follows the Principle of Least Privilege because it returns only the minimal data required by the calling code, rather than providing access to the entire user object. This helps to reduce the risk of data exposure or manipulation.

  - Not following the principle of Least privilege

```
function getUser(id) {
  //fetching a user object in a DB
  const user = DB.findUser(id);
  if (user)
    return user;

  return null;
}
```

In this code example, the getUser function retrieves user data from a database or API based on the id parameter, and then returns the entire user object. This code does not follow the Principle of Least Privilege because it returns more data than necessary, potentially exposing sensitive information to unauthorized parties. If an attacker gains control of this function, they could potentially view or manipulate any aspect of the user's data, which could lead to serious security issues.

- Part 2
  - Principle of fail-safe defaults: following

```
//part 2.1

function validateUserInput(input) {
  // Assume this function validates user input
  if (input==valid) {
   return "Valid";
  }
  return '';
 }
```

In this code example, the validateUserInput function checks whether the input parameter is valid, it returns the “Valid” value. Otherwise, it returns an empty string. This approach follows the Principle of Fail-safe Defaults because it assumes that any input other than a valid non-empty string is potentially unsafe and defaults to an empty string, which is a safe value. This helps to reduce the risk of unexpected behavior or security issues.

  - Not following:

```
//part 2.2

function validateUserInput(input) {
  // Assume this function validates user input
  if (input==valid) {
   return "Valid";
  }
  return null;
 }
```

In this code example, the validateUserInput function checks whether the input parameter is valid, it returns the “valid” value. Otherwise, it returns null. This code does not follow the Principle of Fail-safe Defaults because it assumes that null is a safe value to return in case of an invalid input, which is not necessarily the case. In some cases, returning null could lead to unexpected behavior or security issues. Another example is of switch statements, where in order to prevent unusual behavior, we add a `default` case alongside other cases. If `default` case is not defined, the system may behave unpredictably.

- Part 3 : Principle of economy of mechanism
    - Following:

```
function sumArray(array) {
  let sum = 0;
  for (let i = 0; i < array.length; i++) {
    sum += array[i];
  }
  return sum;
}
```

In this code example, the sumArray function takes an array of numbers and calculates their sum using a simple for loop. The implementation is straightforward and easy to understand, without any unnecessary complexity or components. This approach follows the Principle of Economy of Mechanism because it uses a simple and small mechanism to achieve the desired result.

    - Not Following

```
function calculateAverage(array) {
  let sum = 0;
  for (let i = 0; i < array.length; i++) {
    sum += array[i];
  }
  let average = sum / array.length;
  let roundedAverage = Math.round(average * 100) / 100;
  return roundedAverage;
}
```

In this code example, the calculateAverage function also takes an array of numbers and calculates their average. However, in this case, the implementation includes unnecessary complexity, such as rounding the average to two decimal places. While this feature may be useful in some cases, it is not essential for the function's primary purpose, which is to calculate the average. This approach does not follow the Principle of Economy of Mechanism because it uses a more complex and less straightforward mechanism to achieve the same result.

- Part 4: Principle of complete mediation
    - Following:

```
// Part 4.1

function sensitiveOperation() {
  if (isAuthorized(user)) {
    // perform sensitive operation
  } else {
    // access denied
  }
}

function isAuthorized(user) {
  // check user's permissions and access controls
  // return true if authorized, false otherwise
}
```

In this example, the sensitiveOperation function performs a sensitive operation, and access to this operation is mediated by the isAuthorized function. Before the sensitive operation is performed, the isAuthorized function checks the user's permissions and access controls to ensure that the user is authorized to perform the operation. This approach follows the Principle of Complete Mediation because access to the sensitive operation is only granted after proper authorization has been verified.

- Not following:

```
function sensitiveOperation() {
  // perform sensitive operation
 }
```

In this example, the sensitiveOperation function performs a sensitive operation without any access control mechanism to verify that the user is authorized to perform the operation. This approach does not follow the Principle of Complete Mediation because access to the sensitive operation is not mediated by an access control mechanism.

- Part 5: Principle of separation of privileges:
  - Following:

```
//  part 5.1

function handleUserRequest(user) {
  if (user.isAdmin) {
    // perform admin tasks
  } else {
    // perform user tasks
  }
}
```

In this example, the handleUserRequest function handles requests from users and checks whether the user is an admin or not. If the user is an admin, the function performs administrative tasks that require elevated privileges. If the user is not an admin, the function performs tasks that only require user privileges. This approach follows the Principle of Separation of Privileges because the function limits the privileges of each user to only what they need to perform their tasks.

  - Not Following:

```
function handleUserRequest(user) {
  // perform tasks
  if (user.isAdmin) {
   // perform admin tasks
  }
 }
```

In this example, the handleUserRequest function performs both user and admin tasks in the same function. If the user is an admin, the function performs additional admin tasks without separating the privileges associated with these tasks. This approach does not follow the Principle of Separation of Privileges because it does not limit the privileges associated with different tasks to only what is necessary.

- Part 6: Principle of Least common Mechanism
  - Following:

```
//Part 6.1
function encryptData(data, key) {
   // use a separate encryption function
   const encryptedData = encryptWithKey(data, key);
   return encryptedData;
}

function decryptData(encryptedData, key) {
   // use a separate decryption function
   const decryptedData = decryptWithKey(encryptedData, key);
   return decryptedData;
}
```

In this example, the encryptData and decryptData functions use a separate encryption and decryption function to protect sensitive data. By using a separate mechanism for encryption and decryption, this approach follows the Principle of Least Common Mechanism because it reduces the number of mechanisms that can be affected in the event of a security breach. If the encryption function is compromised, it will not necessarily compromise the decryption function, and vice versa.

- Not Following

```
//Part 6.2
function encryptAndDecryptData(data, key) {
   // use the same mechanism for encryption and decryption
   const encryptedData = encrypt(data, key);
   const decryptedData = decrypt(encryptedData, key);
   return decryptedData;
 }
```

In this example, the encryptAndDecryptData function uses the same mechanism for both encryption and decryption. This approach does not follow the Principle of Least Common Mechanism because it increases the number of mechanisms that can be affected in the event of a security breach. If the encryption function is compromised, it can be used to compromise the decryption function, and vice versa.

## Question#2

1. Hardware security module

Economy of Mechanism: The HSMs gets the principle applied in a way that the design of hardware security modules should be as simple as possible, with minimal components and functionality, to reduce the potential attack surface and make the system easier to secure and maintain
Fail-Safe Defaults: This principle refers to the diminishing of sensitive information (cryptographic) when a security threat is encountered.
Complete Mediation: Furthermore, the principle of "complete mediation" is applicable, as it implies that all access to a resource should be checked and authorized before being allowed, which is critical for protecting the sensitive data encrypted functions and data stored within a hardware security module.

2. Cuckoo sandbox for malware analysis

Complete Mediation: In order to prevent the malware from taking any unintended activities, the Cuckoo sandbox should buffer all interactions between the malware and the system being examined.
Least Privilege: Each component and user should be provided with the bare minimum of access required for them to carry out their respective tasks in the Cuckoo sandbox. This aids in limiting the harm that could be done by a user or component that has been compromised.
Separation of Privilege: The privileges of various components and users should be constrained by the Cuckoo sandbox's design in order to prevent any one component or user from carrying out unauthorized actions.
Least Common Mechanism: To reduce the risk of vulnerabilities, the Cuckoo sandbox should be built with the least number of shared resources possible between components and users.
Economy of mechanism: The Cuckoo sandbox be designed with as little complexity as possible to perform its function. Unnecessary complexity can introduce flaws and make the system more difficult to secure.
Fail-Safe Default: The Cuckoo sandbox should be in a secure state regardless of whether the user actively configures it that way. For instance, by default, the sandbox shouldn't run untested code on the host computer.

3. Access control list in an operating system

Complete Mediation: To guarantee that only authorized users have access to resources, each access attempt should be compared to the access control list.
Least Privilege: Instead of giving users complete access to the system, only the resources they need to carry out their jobs should be made available to them.

Separation of Privilege: To ensure that unauthorized users cannot access resources, the access control list should demand various types of authentication, such as a password and a security token.
Least Common Mechanism: There shouldn't be any shared or attack-prone procedures used in the access control list.
Economy of mechanism: This idea suggests that the system should be designed in a simple and efficient way so that the unwanted complexity that causes security exploits can be minimized.
Fail-safe defaults: The access control list should have secure default administrative setting so that users information isn't compromised.

4. Image Captcha on Flex

Complete Mediation: Every access should be checked and validated to assure that the user has authorized access to the system.
Least Privilege: Each user should be given the minimum amount of access necessary to perform their tasks, to minimize the risk of attacks.
Separation of Privilege: To prevent any one user from having too much authority, the system should require many levels of authorization to access sensitive information or carry out sensitive operations.
Least Common Mechanism: The system should avoid access to shared resources and mechanisms in order to avoid any security threat.
Economy of mechanism: To minimize the danger of vulnerabilities and mistakes, the image captcha system should be as straightforward as feasible.
Fail-safe defaults: The system's default settings must be secure in order to prevent any attack and if happens the system remains secure.

5. Password strength indicator on Google or similar websites when you create an

Account
Complete Mediation: This principle refers to the creation and enforcement of having strong passwords so that the threats can be minimized.
Least Privilege: The password strength indicator should be made to only grant users the minimal amount of access required for them to carry out their intended functions.
Separation of Privilege: The password strength indicator should have a distinct separation of privileges to guarantee that only individuals with the proper authorization can access critical information
Least Common Mechanism: The password strength indicator should be designed in a way that it doesn't compromise the security breaches hence, minimizing shared resources and setups.
Economy of mechanism: It should be designed in a way that the functionality can be achieved in an efficient manner without facing any unnecessary complexity.
Fail-safe defaults: The system should be configured with the password strength indicator's default settings to make sure that it is secure by default and that any potential vulnerabilities are minimized.

6. Biometric authentication required before using banking app

Complete Mediation: The biometric authentication should go through a safe authentication method to make sure that only permissible users are registered.
Least Privilege: This refers to the fact that users should be provided with necessary level of details required for them to fulfill their tasks, minimizing the vulnerability due to unauthorized access.
Separation of Privilege: To prevent unauthorized access to sensitive data, biometric authentication systems should have different degrees of permission, such as administrators and users.
Least Common Mechanism: The biometric authentication should involve unique mechanisms for every task in order to attain security such as data collection is done in a secure way to avoid the risk of data breaches.
Economy of mechanism: This means that the biometric authentication systems should be designed in a simple but efficient manner to minimize threats.
Fail-safe defaults: These systems should have fail-safe defaults feature so that if system fails, no unauthorized access takes place.

7. The way encryption ciphers like AES were designed (look at the history of AES

first)
Complete Mediation: Every access to encrypted data should be permitted and properly validated in ciphers.
Least Privilege: In this principle, the encryption key is kept secure so that no unauthorized activity takes place.
Separation of Privilege: The encryption should make sure that different individuals or groups are given access at various levels according to the least privilege concept.
Least Common Mechanism: The cipher should assure that its specific mechanisms are not to be shared with other security systems.
The rest two principles i.e. fail-safe defaults and economy of mechanism are similar as discussed in earlier systems.

8. Atomicity in database transactions

Economy of mechanism: The level of atomicity in the system should be simple to understand so that vulnerabilities can be minimized.

Complete mediation: Every transactions should be checked for atomicity, and there should be no exceptions or vulnerable points in the atomicity enforcement process. Transaction management systems, which guarantee the consistency of database updates, can be used to do this.
Least privilege:  Database transactions should only be accessible to those who need them, and each user should only have the bare minimum of privileges. This can lessen the likelihood of data breaches and assist prevent unwanted access.

9. Intrusion detection systems in front of public facing servers of an organization:

Complete mediation: Every entry into the system needs to be verified and checked. The intrusion detection system ought to be created to prevent all unwanted access, including that from insiders and malicious actors.
Separation of privileges: Users should only be given the minimal access rights required to operate the system, and these rights should be delegated in accordance with the user's role and duties.
Least privilege: The system should function with the minimal rights required to carry out its tasks. This idea aids in stopping malicious actors' unwanted access and damage.
Least common mechanism: The system should use the least common mechanisms possible so that attackers cannot take advantage of them. For instance, shared passwords or credentials shouldn't be used by intrusion detection systems.
Fail-safe defaults: In the event of failure or human error, the system must to have secure and dependable default settings. Systems for detecting intrusions, for instance, should contain default options that reduce the number of false positives and false negatives.

## Question#3:

The least privilege principle is violated by the given CVE-2007-0408. According to this idea, each application and user should utilize the least rights essential to perform their tasks. However, CVE-2007-0408 enables attackers to take advantage of a Windows operating system exploit to gain administrative privileges on a system. Sensitive information may be compromised as a result of this violation, and system configurations may be changed without authorization.

## Question#04

The Principle of Least Common Mechanism is a security principle that is concerned with reducing the potential impact of security vulnerabilities. The principle states that a system should use the least number of mechanisms possible, and each mechanism should be as simple as possible. The idea is to reduce the impact of a security breach by minimizing the number of mechanisms that can be affected by the breach.

1. Air-gapping of important machines/servers in companies:

The Principle of Least Common Mechanism is enforced in this scenario, as air-gapping involves physically isolating important machines or servers from the public internet or other networks. By doing this, the number of mechanisms that can be affected in the event of a security breach is reduced, as

there is no direct connection between the isolated machines and external networks. This approach also ensures that sensitive data cannot be accessed or manipulated through common mechanisms, such as software vulnerabilities, malware or unauthorized network connections.
In simple words, this principle is followed by physically isolating them from external networks, making it difficult for attackers to breach the system.

2. Cloudflare protection for websites:

The Principle of Least Common Mechanism is enforced in this scenario, as Cloudflare's security services use a variety of mechanisms to protect websites from various types of attacks, including DDoS attacks, SQL injection, and cross-site scripting attacks. Cloudflare's services distribute incoming web traffic across a global network of servers, while also providing a layer of security, such as SSL/TLS encryption, web application firewall, rate limiting, and bot mitigation. By using a combination of mechanisms, Cloudflare ensures that a single mechanism is not responsible for protecting websites from all types of attacks, reducing the risk of compromise.
In simple words, the principle is followed by using a variety of security mechanisms to protect websites from different types of attacks, reducing the risk of a single mechanism being compromised.

3. The Colonial Pipeline ransomware attack:

The Principle of Least Common Mechanism is violated in this scenario, as the Colonial Pipeline's IT systems were not properly segregated, and a single VPN account was used to access multiple systems. This approach increased the number of mechanisms that can be affected in the event of a security breach, as the compromise of a single mechanism could allow the attacker to move laterally through the system and gain access to other sensitive systems. By not properly segregating their IT systems and using a single account for multiple systems, the Colonial Pipeline created a single point of failure, which was exploited by the ransomware attackers.
In simple words, the organization failed to properly segregate its IT systems and used a single account for multiple systems, which increased the risk of a security breach and allowed the attackers to move laterally through the system and gain access to other sensitive systems.

Overall, the Principle of Least Common Mechanism emphasizes the importance of minimizing the number of mechanisms that can be affected in the event of a security breach. By following this principle, organizations can reduce the impact of a security breach and limit the scope of damage that can be caused.