

## A.Information Security Basics

**Confidentiality:** This is all about keeping secrets safe. Imagine information as a treasure chest. Confidentiality ensures only authorized people have the key to unlock it and see what's inside. This could involve encrypting data, controlling access to files, and being careful about what information we share.

Analogy :Confidentiality (Secret): It's like a whisper! Only your friend should be able to hear it. You wouldn't want someone else to find out about the surprise party! In information security, this means keeping your information private, like passwords or secret codes.

**Integrity:** This pillar focuses on making sure information is accurate and hasn't been tampered with. Think of it like a recipe. Integrity ensures the recipe hasn't been changed, so the final dish turns out as expected. Security measures like checksums and digital signatures help maintain data integrity, preventing unauthorized modifications.


Analogy: Integrity (Accurate): It's like making sure the message doesn't get messed up on the way. You wouldn't want your friend to hear "bring carrots" instead of "bring cupcakes" for the party! In information security, this means making sure information stays accurate and hasn't been changed by someone who shouldn't have.

**Availability:** This means authorized users can access the information they need, whenever they need it. Imagine a library. Availability ensures the library is open during operating hours, and the books you need are on the shelf, ready for you to borrow. Security measures like backups and system redundancy help maintain availability, preventing outages that could lock people out of essential information.

Analogy :Availability (Accessible): It's like being able to deliver the message. Your friend needs to get your message so they can come to the party! In information security, this means making sure information is accessible when you need it. You wouldn't want your friend to lose the message and miss out on the fun!


## B.Application Security

- The code that vuln with sql injection

Php  Copy code

```
$user_login =  
trim($_POST["user_login"]);  
...  
$sql = "SELECT id, username,  
password FROM users WHERE  
username = '$user_login' OR  
email = '$user_login'";
```

- The code after fix Sql injection

Php  Copy code

```
$user_login =  
mysqli_real_escape_string($link,  
$_POST["user_login"]);
```

Php

 Copy code

```
$sql = "SELECT id, username,  
password FROM users WHERE  
username = ? OR email = ?";
```

To prevent SQL injection attacks:

1. Use Parameterized Queries or Prepared Statements: Instead of directly inserting user input into SQL queries, use parameterized queries or prepared statements. They separate user input from SQL commands, making it impossible for attackers to inject malicious code.

Example in PHP:

```
$stmt = $pdo->prepare("SELECT id, username, password FROM users  
WHERE username = :username OR email = :email");
```

```
$stmt->execute(['username' => $user_login, 'email' => $user_login]);
```

2. Input Validation: Validate and sanitize user input to ensure it matches expected formats and patterns, further reducing the risk of malicious input.

3. Limit Database Permissions: Restrict database permissions for application users to minimize the impact of potential SQL injection attacks.

## C.Cloud Security

In cloud security, unlike a physical setup where everything is your concern, responsibility is divided between you and the cloud provider. It's like sharing an apartment.

The cloud provider takes care of the building security (physical security, overall infrastructure) just like the landlord looking after the building itself. But you're responsible for securing your own stuff inside the apartment (your data, applications, configurations) and making sure you lock your doors (access controls). This shared responsibility model makes cloud security a team effort.

## D.Git Knowledge

1.Git, it's like having a magic notebook when you can:

- Saves Drafts: Every time you finish a bit of your story, Git saves it like a draft. You can see all your drafts and pick the one you like best.
- Undo Mistakes: Accidentally wrote something weird? No sweat! Just go back to an earlier draft and fix it.
- Work with Friends: Want a friend to help write the story? Git lets you both work on the story at the same time, without messing each other's work up.
- Never Lose Your Work: Even if you spill juice on your notebook (or your computer crashes!), Git keeps a safe copy of all your drafts.

2.A git commit message is a brief description of the changes you've made to your code that are being captured in a particular commit. It helps you and others understand what was modified and why.

Here are some key points about writing good commit messages:

Structure: It typically consists of two parts: a subject line (50 characters or less) summarizing the change, followed by a more detailed description (optional) separated by a blank line.

Subject Line:

Start with a verb in the imperative mood (e.g., "Fix", "Add", "Update").

Capitalize the first word and avoid ending with a period.

Keep it concise and informative.

Body (Optional):

Provide more context about the change, like why it was made.

Use bullet points for clarity if needed.

Wrap text at 72 characters.

Here's an example:

Subject: Fix typo in documentation (#123) it's mean line 123

Updated the README to correct a spelling error in the installation instructions.

This message clearly indicates that a typo in the documentation was fixed, while also referencing a relevant issue number (if applicable).

3.Git Issue

**F.Team Work**

1.To manage schedule first i priorities task that important and urgent,Usually I write down and make a schedule that will be done tomorrow morning at night before going to bed

2.The way I communicate with my colleagues is that I make a schedule first and ask when they have free time, then continue to contact me at the agreed time. If I am unable to do so, I will contact them via email for formal assignments.

3.I set my priorities by first understanding the overarching goals and objectives. Then, I assess tasks based on their urgency, importance, and alignment with those goals. I use techniques like prioritization matrices to allocate time and resources effectively, ensuring that I focus on high-impact activities. Additionally, I regularly review and adjust priorities as needed to stay flexible and responsive to changing circumstances.

4.I trust my team by communicating openly, empowering them to make decisions, providing support when needed, leading by example, giving feedback regularly, and recognizing their contributions.

5.When giving feedback to my team, I keep it simple: Be specific, timely, and respectful. I focus on behaviors, offer constructive suggestions, and encourage open dialogue for improvement.

## F.Project

Link : <https://github.com/Gen0z/ITSec>

To fix or mitigate the SQL injection vulnerability in the code, you should use parameterized queries or prepared statements instead of directly concatenating user input into the SQL query string. Here's how you can do it:

Replace this line:

```
$sql = "SELECT id, username, password FROM users WHERE username =  
'$user_login' OR email = '$user_login'";
```

With prepared statements using mysqli:

```
$sql = "SELECT id, username, password FROM users WHERE username = ?  
OR email = ?";
```

```
$stmt = mysqli_prepare($link, $sql);
```

```
if ($stmt) {
```

```
    mysqli_stmt_bind_param($stmt, "ss", $user_login, $user_login);
```

```
    mysqli_stmt_execute($stmt);
```

```
    $result = mysqli_stmt_get_result($stmt);
```

```
    // Continue with the rest of your code
```

```
} else {
```

```
    // Handle error
```

}

Using prepared statements will automatically sanitize user input and prevent SQL injection attacks by treating user input as data rather than part of the SQL query structure. This approach ensures that user input is safely escaped and does not interfere with the logic of the SQL query.