## Task 9 : 3. Injection

### Injection

Injection flaws are very common in applications today. These flaws occur because the application interprets user-controlled input as commands or parameters. Injection attacks depend on what technologies are used and how these technologies interpret the input. Some common examples include:

- **SQL Injection:** This occurs when user-controlled input is passed to SQL queries. As a result, an attacker can pass in SQL queries to manipulate the outcome of such queries. This could potentially allow the attacker to access, modify and delete information in a database when this input is passed into database queries. This would mean an attacker could steal sensitive information such as personal details and credentials.
- **Command Injection:** This occurs when user input is passed to system commands. As a result, an attacker can execute arbitrary system commands on application servers, potentially allowing them to access users' systems.

The main defence for preventing injection attacks is ensuring that user-controlled input is not interpreted as queries or commands. There are different ways of doing this:

- **Using an allow list:** when input is sent to the server, this input is compared to a list of safe inputs or characters. If the input is marked as safe, then it is processed. Otherwise, it is rejected, and the application throws an error.
- **Stripping input:** If the input contains dangerous characters, these are removed before processing.

Dangerous characters or input is classified as any input that can change how the underlying data is processed. Instead of manually constructing allow lists or stripping input, various libraries exist that can perform these actions for you.

## Answer the questions below :

1. I've understood Injection attacks.
A. No answer needed