Cross Site Scripting (XSS)

→ Cross Site Scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website.

→ The actual attack occurs when the victim visits the web application that executes the malicious code.

→ A web page or web application is vulnerable to XSS if it uses unsanitized user input in the output that it generates.

→ This user input must then be parsed by the victim's browser.

→ XSS attacks are possible in VBScript, ActiveX, Flash, and even CSS. However they are common in Javascript, because Javascript is fundamental to most browsing experiences.

How Cross-site Scripting works

There are 2 stages to a typical XSS attacks.

1) To run malicious Javascript code in a victim's browser, an attacker must first find a way to inject malicious code into a web page that the victim visits.

2) After that, the victim must visit the web page with the malicious code. If the attack is directed at particular victims, the attacker can use social engineering and/or phishing to send a malicious URL to the victim.

SQL Injection

→ SQL injection is a code injection technique that might destroy the database.

→ It is most common web hacking technique, allow attacker to view data that they are not able to retrieve.

→ SQL injection is the placement of malicious code in SQL statements, via web page input. (web security Vulnerability that allow attacker to interfare with the queries.)

SQL in Web pages

→ SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives us an SQL statement that we will unknowingly run on our databases.

Example →

```
txtUserId =
getRequestString ("UserID");
txtSQL = "SELECT * FROM Users WHERE
UserID = " + txtUserID;
```

above example creates a Select statement by adding a Variable (txtUserID) to a Select String. The variable is fetched from user input (getrequestString)

Some Common SQL injection example include :-

→ Retrieving hidden data, where you can modify an SQL query to return additional results.

→ Subverting application logic, where we can change a query to interface with the applications logic.

→ Union attacks, where we can retrieved data from different database tables.

→ Examining the databases, where we can extract information about version and structure of the database.

→ Blind SQL injection, where the results of a query we control are not returned in the applications responses.

How to detect SQL Injection Vulnerabilities

→ SQL injection vulnerabilities can be found using web vulnerability scanner and manually by using a systematic set of tests against every entry point in the application