🔐 1. SQL Injection Testing (Selenium + TestNG)

🎯 Purpose

To check if application fields (login, search, form fields) accept harmful SQL commands.

□ Selenium + TestNG Script

```
@Test
public void testSQLInjection() {
    WebDriver driver = new ChromeDriver();
    driver.get("https://example.com/login");

    driver.findElement(By.id("username")).sendKeys("' OR '1'='1");
    driver.findElement(By.id("password")).sendKeys("abc");
    driver.findElement(By.id("loginBtn")).click();

    boolean isAlert = false;
    try {
        driver.switchTo().alert();
        isAlert = true;
    } catch (Exception e) {}

    Assert.assertFalse(isAlert, "SQL Injection Vulnerability Found!");
    driver.quit();
}
```

🛡 2. XSS (Cross-Site Scripting)

🎯 Purpose

To check if input fields allow JavaScript execution.

□ Selenium + TestNG Script

```
@Test
public void testXSS() {
    WebDriver driver = new ChromeDriver();
    driver.get("https://example.com/comment");


driver.findElement(By.id("comment")).sendKeys("<script>alert('XSS')</script
>");
    driver.findElement(By.id("submit")).click();

    try {
        driver.switchTo().alert();
        Assert.fail("XSS alert triggered! Vulnerability detected.");
    } catch (NoAlertPresentException e) {
        Assert.assertTrue(true);
    }
    driver.quit();
}
```

🔒 3. CSRF (Cross-Site Request Forgery)

🎯 Purpose

To ensure sensitive operations require a **CSRF token** and cannot be executed directly.

□ Selenium + TestNG Script

```
@Test
public void testCSRFProtection() {
    WebDriver driver = new ChromeDriver();
    driver.get("https://example.com/login");

    driver.findElement(By.id("username")).sendKeys("test");
    driver.findElement(By.id("password")).sendKeys("pass");
    driver.findElement(By.id("loginBtn")).click();

    // Trying to execute an operation without CSRF token
    driver.get("https://example.com/change-password?pwd=hacked");

    Assert.assertTrue(
        driver.getPageSource().contains("Invalid CSRF")
        || driver.getCurrentUrl().contains("error"),
        "CSRF protection failed!"
    );

    driver.quit();
}
```

🔑 4. Password Encryption Testing

🎯 Purpose

To make sure passwords are not stored or visible in browser memory or API response.

□ Selenium Script (Checks browser storage)

```
@Test
public void testPasswordNotInLocalStorage() {
    WebDriver driver = new ChromeDriver();
    driver.get("https://example.com/login");

    JavascriptExecutor js = (JavascriptExecutor) driver;
    String storedPassword = (String) js.executeScript(
        "return window.localStorage.getItem('password');"
    );

    Assert.assertNull(storedPassword, "Password stored in LocalStorage —
Security Risk!");
    driver.quit();
}
```

🔐 5. HTTPS Enabled Testing

🎯 Purpose

To verify that application uses HTTPS for secure communication.

Selenium Script

```
@Test
public void testHTTPSEnabled() {
    WebDriver driver = new ChromeDriver();
    driver.get("https://example.com");

    String url = driver.getCurrentUrl();
    Assert.assertTrue(url.startsWith("https"), "HTTPS is not enabled!");

    driver.quit();
}
```

---

6. Session Timeout Testing

🎯 Purpose

To ensure idle sessions are invalidated after timeout (ex: 10 minutes).

Selenium + TestNG Script

```
@Test
public void testSessionTimeout() throws InterruptedException {
    WebDriver driver = new ChromeDriver();
    driver.get("https://example.com/login");

    driver.findElement(By.id("username")).sendKeys("user");
    driver.findElement(By.id("password")).sendKeys("pass");
    driver.findElement(By.id("loginBtn")).click();

    // Wait for session expiration time (example: 10 minutes)
    Thread.sleep(600000); // 10 minutes

    driver.navigate().refresh();

    Assert.assertTrue(
        driver.getCurrentUrl().contains("login"),
        "Session timeout NOT working!"
    );
    driver.quit();
}
```