**Some key terms used in security testing**

Before we go further, it will be useful to be aware of a few terms that are frequently used in web application security testing:

**What is “Vulnerability”?**  
This is a weakness in the web application. The cause of such a “weakness” can be bugs in the application, an injection (SQL/ script code) or the presence of viruses.  
  
**What is “URL manipulation”?**  
Some web applications communicate additional information between the client (browser) and the server in the URL. Changing some information in the URL may sometimes lead to unintended behavior by the server.

**What is “SQL injection”?**  
This is the process of inserting SQL statements through the web application user interface into some query that is then executed by the server.

**What is “XSS (Cross Site Scripting)”?**  
When a user inserts HTML/ client-side script in the user interface of a web application and this insertion is visible to other users, it is called XSS.

**What is “Spoofing”?**  
The creation of hoax look-alike websites or emails is called spoofing.  
**Security testing approach**:

In order to perform a useful security test of a web application, the security tester should have good knowledge of the HTTP protocol. It is important to have an understanding of how the client (browser) and the server communicate using HTTP. Additionally, the tester should at least know the basics of SQL injection and XSS. Hopefully, the number of security defects present in the web application will not be high. However, being able to accurately describe the security defects with all the required details to all concerned will definitely help.

**1. Password cracking:**

The security testing on a web application can be kicked off by “password cracking”. In order to log in to the private areas of the application, one can either guess a username/ password or use some password cracker tool for the same. Lists of common usernames and passwords are available along with open source password crackers. If the web application does not enforce a complex password (e.g. with alphabets, number and special characters, with at least a required number of characters), it may not take very long to crack the username and password.

If username or password is stored in cookies without encrypting, attacker can use different methods to steal the cookies and then information stored in the cookies like username and password.

For more details see article on “[Website cookie testing](http://www.softwaretestinghelp.com/website-cookie-testing-test-cases/)”.

**2. URL manipulation through HTTP GET methods:**

The tester should check if the application passes important information in the querystring. This happens when the application uses the HTTP GET method to pass information between the client and the server. The information is passed in parameters in the querystring. The tester can modify a parameter value in the querystring to check if the server accepts it.

Via HTTP GET request user information is passed to server for authentication or fetching data. Attacker can manipulate every input variable passed from this GET request to server in order to get the required information or to corrupt the data. In such conditions any unusual behavior by application or web server is the doorway for the attacker to get into the application.

**3. SQL Injection:**

The next thing that should be checked is SQL injection. Entering a single quote (‘) in any textbox should be rejected by the application. Instead, if the tester encounters a database error, it means that the user input is inserted in some query which is then executed by the application. In such a case, the application is vulnerable to SQL injection.

SQL injection attacks are very critical as attacker can get vital information from server database. To check SQL injection entry points into your web application, find out code from your code base where direct MySQL queries are executed on database by accepting some user inputs.

If user input data is crafted in SQL queries to query the database, attacker can inject SQL statements or part of SQL statements as user inputs to extract vital information from database. Even if attacker is successful to crash the application, from the SQL query error shown on browser, attacker can get the information they are looking for. Special characters from user inputs should be handled/escaped properly in such cases.

**4. Cross Site Scripting (XSS):**

The tester should additionally check the web application for XSS (Cross site scripting). Any HTML e.g. <HTML> or any script e.g. <SCRIPT> should not be accepted by the application. If it is, the application can be prone to an attack by Cross Site Scripting.

Attacker can use this method to execute malicious script or URL on victim’s browser. Using cross-site scripting, attacker can use scripts like JavaScript to steal user cookies and information stored in the cookies.

Many web applications get some user information and pass this information in some variables from different pages.

E.g.: http://www.examplesite.com/index.php?userid=123&query=xyz

Attacker can easily pass some malicious input or <script> as a ‘&query’ parameter which can explore important user/server data on browser.

**Important:** During security testing, the tester should be very careful not to modify any of the following:

* Configuration of the application or the server
* Services running on the server
* Existing user or customer data hosted by the application

Additionally, a security test should be avoided on a production system.

The purpose of the security test is to discover the vulnerabilities of the web application so that the developers can then remove these vulnerabilities from the application and make the web application and data safe from unauthorized actions.