SQL INJECTION

Muhamed Medhat . DevTeam

INTRO

SQL Injection

is a method of exploiting the database of web application. It is done by injecting the SQL statements as an input string to gain an unauthorized access to a database.

SQL injection is a serious vulnerability that leads to a high level of compromise - usually the ability to run any database query.

It is a web-based application attack that connects to database back-ends and allows the bypass of firewall.

The advantage of insecure code and bad input validation is clinched by the attacker to execute unauthorized SQL commands. The objective of SQL Injection Attack (SQLIA) is to shaft the database system into running harmful code that can reveal confidential information.

Hence we can say that Sql injection attack is an unauthorized access of database.

Understanding How Web Applications Work

Most of us use Web applications on a daily basis, either as part of our vocation or in order to access our e-mail, book a holiday, purchase a product from an online store, view a news item of interest, and so forth.

One thing that Web applications have in common, regardless of the language in which they were written, is that they are **interactive and, more often than not, are databasedriven.**

They normally consist of a back-end database with Web pages that contain server-side script written in a programming language that is capable of extracting specific information from a database depending on various dynamic interactions with the user.

A database-driven Web application commonly has three tiers: a presentation tier (a Web browser or rendering engine), a logic tier (a programming language, such as C#, ASP, .NET, PHP, JSP, etc.), and a storage tier (a database such as Microsoft SQL Server, MySQL, Oracle, etc.).

The Web browser (the presentation tier, such as Internet Explorer, Safari, Firefox, etc.) sends requests to the middle tier (the logic tier), which services the requests by making queries and updates against the database (the storage tier).

for example,

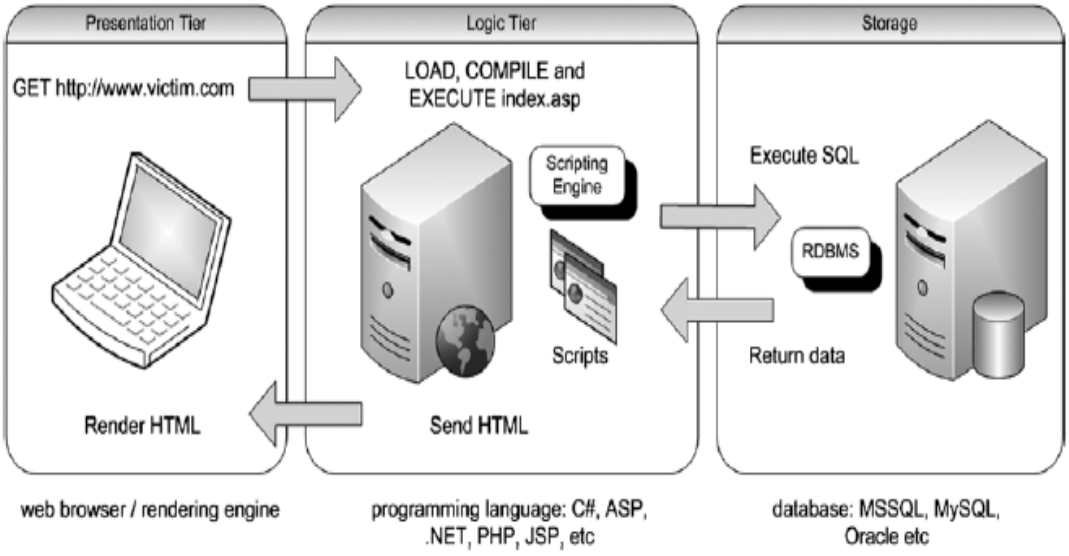
an online retail store that presents a search form that allows you to sift and sort through products that are of particular interest, and provides an option to further refine the products that are displayed to suit financial budget constraints. To view all products within the store that cost less than $100, you could use the following URL:

 http://www.victim.com/products.php?val=100

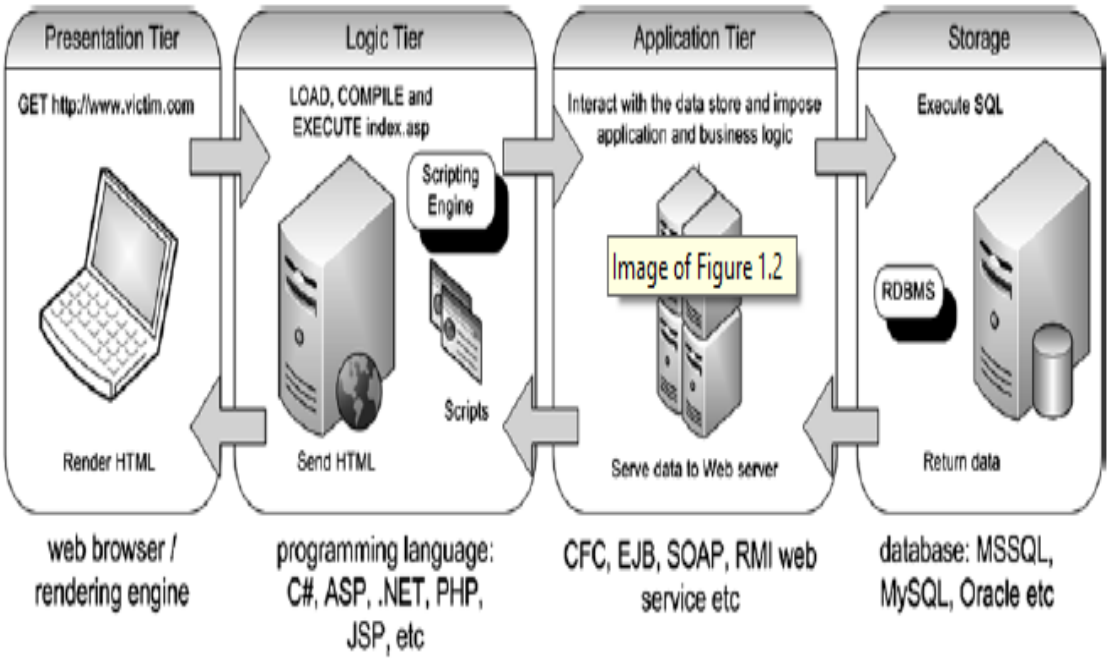
The following PHP script illustrates how the user input (*val*) is passed to a dynamically created SQL statement. The following section of the PHP code is executed when the URL is requested.

```
// connect to the database

$conn = mysql_connect("localhost","username","password");

// dynamically build the sql statement with the input

$query = "SELECT * FROM Products WHERE Price < '$_GET["val"]' " .

"ORDER BY ProductDescription";

// execute the query against the database

$result = mysql_query($query);

// iterate through the record set

while($row = mysql_fetch_array($result, MYSQL_ASSOC))

{// display the results to the browser

echo "Description : {$row['ProductDescription']} <br>" .

"Product ID : {$row['ProductID']} <br>" .

"Price : {$row['Price']} <br><br>";

}
```

Simple Three-Tier Architecture



More Complex Architecture



Understanding SQL Injection

SQL injection is an attack in which SQL code is inserted or appended into application/user input parameters that are later passed to a back-end SQL server for parsing and execution.

Any procedure that constructs SQL statements could potentially be vulnerable, as the diverse nature of SQL and the methods available for constructing it provide a wealth of coding options.

The primary form of SQL injection consists of direct insertion of code into parameters that are concatenated with SQL commands and executed.

A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata.

When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed. When a Web application fails to properly sanitize the parameters which are passed to dynamically created SQL statements (even when using parameterization techniques) it is possible for an attacker to alter the construction of back-end SQL statements.

When an attacker is able to modify an SQL statement, the statement will execute with the same rights as the application user; when using the SQL server to execute commands that interact with the operating system, the process will run with the same permissions as the component that executed the command (e.g., database server, application server, or Web server), which is often highly privileged.

returning to the previous example of a simple online retail store

This time, however, you are going to attempt to inject SQL commands by appending them to the input parameter *val*. You can do this by appending the string

‘ OR ‘1’= ‘1 **to the URL:**

■■ http://www.victim.com/products.php?val=100’ OR ‘1’=’1

This time, the SQL statement that the PHP script builds and executes will return all of the products in the database regardless of their price. This is because you have altered the logic of the query. This happens because the appended statement results in the *OR* operand of the query always returning *true*, that is, 1 will always be equal to 1. Here is the query that was built and executed:

```
SELECT *
FROM ProductsTbl
WHERE Price < '100.00' OR '1'='1'
ORDER BY ProductDescription;
```

END