Journal of Advanced Computing and Communication Technologies ***(ISSN:* 2347 - 2804*)***
Volume No 4 Issue No2, April 2016

ANALYSIS OF VARIOUS LEVELS OF PENETRATION BY SQL INJECTION TECHNIQUE THROUGH DVWA

By

Ashish Kumar, Dr. Swapnesh Taterh
1st AIIT, Amity University, Rajasthan.
2nd Asst Prof, AIIT, Amity University, Rajasthan.
ashishk741@gmail.com, staterh@jpr.amity.edu

Abstract:

In this paper, we represent a comparative analysis of several levels of SQL Injection vulnerabilities, attacks. We are now going to define SQLI. It is very harmful and directly injected into the database SQLI is a technique to access data from the database with the help of SQLI (SQL injection) query. We will define SQL injection process through following procedure:

DVWA (Damn Vulnerable Web Application).Damn Vulnerable Web Application (DVWA) is tool to work on PHP/MySQL. It is facilitated for security professionals and net developers to check out their skills and tools in a legal environment. It may help web developers to know the processes of securing web application. The DVWA tool is used for analyzing the vulnerabilities through SQL Injection.

Keywords
SQL Injection, Penetration, Security

Introduction:

SQLI is a technique to access data on the web, the attacker can access the database on the internet. SQL Injection is a type of attack in a Web application, in which the attacker provides Structured Query Language (SQL) code to a user input box of a Web form to gain unauthorized access data from the database server for the website. SQLI is type of vulnerability often found in web application. By injecting SQL commands, SQL statements can be altered and it compromises the security of a web application. Therefore user may enter some malicious SQL statements that will be placed in the SQL query. SQL query is executed on server side. A successful SQL injection exploits sensitive data from the database, such as username, password.

SQL INJECTION ATTACK:

SQL Injection is a type of attack in a web application, in which the attacker provides SQL code to input box of a web form to gain unauthorized access. Person can access data from the database server from the website. It is method for stealing the data from backend (database), by the help SQLI attack, hacker can get access to the database and steals sensitive information. Generally these attacks are generated from web input so these are called input validation attacks. SQLI can destroy your database.

SQL INJECTION ATTACKS:

Table -1

Category	Description
First Order Attack	The attacker can simply enter a malicious string and cause the modified code to be executed immediately.
Second Order Attack	In second-order injections, attackers inject malicious inputs into a system or database to indirectly trigger an SQLIA, when that input is used at a later time. The objective of this kind of attack differs significantly from a regular (i.e., firstorder) injection attack. example, a user registers on a website using a seeded user name, such as "admin' -- ". The application properly escapes the single quote in the input before storing it in the database, preventing its potentially malicious effect. At this point, the user modifies his or her password, an operation that typically involves (1) checking that the user knows the current password and (2) changing the password if the check is successful. To do this, the Web application might construct an SQL command as follows:

	queryString="UPDATE users SET password='" + newPassword + "' WHERE userName='" + userName + "' AND password='" + oldPassword + "'" newPassword and oldPassword are the new and old passwords, respectively, and userName is the name of the user currently logged-in (i.e., ''admin'--''). Therefore, the query string that is sent to the database is (assume that newPassword and oldPas-sword are "newpwd" and"oldpwd"): UPDATE users SET password='newpwd' WHERE userName= 'admin'' --' AND password='oldpwd' The attacker inserts into persistent storage (such as a table row) which is deemed as a trusted source. An attack is subsequently executed by another activity.
Lateral Injection	The attacker can manipulate the implicit functionTo_Int () by changing the values of the environment variables, Date Format or Numeric Characters.

SQL Injection Method:

Here are some methods through which SQL statements are injected into vulnerable systems.

1. Injected through user input.

In this case, attackers inject SQLcommands by providing suitably crafted user input. A Web application can read user input in several ways based on the environment in which the application is deployed. In most SQLIAs that target Web applications, user input typically comes from form submissions that are sent to the Web application via HTTP GET or POST requests. Web applications are generally able to access the user input contained in these requests as they would access any other variable in the environment.

2. Injection through cookies:

Cookies are files that contain state information generated by Web applications and stored on the client machine. When a client returns to a Web application, cookies can be used to restore the client's state information. Since the client has control over the storage of the cookie, a malicious client could tamper with the cookie's contents. If a Web application uses the cookie's contents to build SQL queries, an attacker could easily submit an attack by embedding it in the cookie.

3. Injection through Server Variables :

Server variables are a collection of variables that contain HTTP, network headers, and environmental variables. Web applications use these server variables in a variety of ways, such as logging usage statistics and identifying browsing trends. If these variables are logged to a database without sanitization, this could create an SQL injection vulnerability .Because attackers can forge the values that are placed in HTTP and network headers, they can exploit this vulnerability by placing an SQLIA directly into the headers. When the query to log the server variable is issued to the database, the attack in the forged header is then triggered.

SQL Injection Level of Security:

Table-2 SQLI security Level

Levels	Description
Low Level	This security level is completely vulnerable and has no security at all. It is easily use by attacker. Example: A' or "='
Medium Level	It is best than low level but this is level have some issue about the security and it is bad security practices for the user. Where the developer has tried but failed to secure an application. It also as a challenge to users to refine their techniques. Example: 1 or 1=1
High Level	It is best and secure level. This level is to give good coding practices for the user. This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.

SQL INJECTION ATTACK TECHNIQUES:

1. Tautologies:

The attacker used SQLI to inject the conditional query statement into the database.
The attacker insert the malicious code into the SQL query, it is based on the condition with the help of this query, attacker can get all the record because where clause always return TRUE value. The attacker by passes the authentication control and accesses the data by exploiting vulnerable input field in which WHERE clause is used.

Purpose of the Tautologies:

- Identify injectable parameters
- Bypass authentication
- Extract data

[1] String SQL Injection:

This type of injection is also referred to as AND/OR Attack. The attacker inputs SQL strings to a conditional query statement and that always evaluates to a true statement in WHERE clause, like '1' OR '1' ='1'.

The goal behind this type of attack is as follows:

- Bypassing authentication,
- Identifying parameters that can be inserted
- Extraction of data.

Scenario

• Normal Statement:
SELECT * FROM login WHERE name='1'
Output :
ID: 1
First name: admin
Surname: admin

[2] SQL Injection Statement:Here numeric value is used instead of strings.
SELECT * FROM loginWHERE name= 1 OR 1 =1.
Output**:** it return all rows from users table.

ID: 1 or 1=1
First name: admin
Surname: admin
ID: 1 or 1=1
First name: Gordon
Surname: Brown
ID: 1 or 1=1
First name: Hack
Surname: Me
ID: 1 or 1=1
First name: Pablo
Surname: Picasso
ID: 1 or 1=1
First name: Bob
Surname: Smith

Piggy-Backed Queries / Statement Injection Attack :

Additional malicious queries are inserted into an original injected query

Purpose :

Extract data
Modify dataset
Execute remote commands
Denial of service

Similar to union queries, this kind of attack appends additional queries to the original query string. If the attack is successful, the database receives and executes a query string that containsmultiple distinct queries..

Example:

SELECT username FROM login WHERE user_name='doe' AND password=''; drop table users -- ' AND pin=234

Introduction of DVWA:

Damn Vulnerable Web Application (DVWA) is tool built using PHP/MySQL. It is an aid for security professionals and Web developers to test and try out their skills and tools in a legal practice environment.
It may help web developers to understand the processes of securing web application. The DVWA tool is used for analyzing the vulnerabilities through SQL Injection.
This tools covers the following topics: Brute Force, Command Execution, File Inclusion, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. Compared to OWASP, it is less comprehensive and covers only few topics. In this tool, there is a lack of adequate information not only of direct topic-related discussions but also of guidelines, hints, and solutions.

Low SQL Injection Source:

```
<?php

if(isset($_GET['Submit'])){

   // Retrieve data

   $name = $_GET['name'];

   $postid = "SELECT user_name, password FROM login WHER
E user_name = '$name'";
   $end = mysql_query($postid) or die('<pre>' . mysql_error() . '</
pre>' );

   $show = mysql_numrows($end);
$i = 0;
  while ($i < $show) {

       $name = mysql_result($end,$i,"user_name");
       $pass = mysql_result($end,$i,"password");
         echo '<pre>';
       echo 'ID: ' . $name. '<Br>user name: ' .$name. '<Br>password
: . $pass;
       echo '</pre>';
```

```
    $i++;
  } }
?>
```

Medium SQL Injection Source:

```
<?php

if (isset($_GET['Submit'])) {

  // Retrieve data

  $name = $_GET['name'];
  $name = mysql_real_escape_string($name);

  $postid = "SELECT user_name, password FROM login WHER
E user_name= $name";
  $end = mysql_query($postid) or die('<pre>' . mysql_error() . '</
pre>' );
  $show = mysql_numrows($end);

  $i=0;

  while ($i < $show {

   $name = mysql_result($end,$i,"user_name");
     $pass = mysql_result($end,$i,"password");
   echo '<pre>';  echo 'ID: ' . $id . '<br>user name: ' . $name . '<br
>password: ' . $pass;
     echo '</pre>';
  $i++;  }
}
?
```

High SQL Injection Source:

```
<?php

if (isset($_GET['Submit'])) {

  // Retrieve data
  $name = $_GET['name'];
  $name = stripslashes($name);

  $name = mysql_real_escape_string($name);
  if (is_numeric($name)){

    $pstid = "SELECT user_name, password
FROM login WHERE user_name = '$name'"; $end = mysql_query
($getid) or die('<pre>'.mysql_error() . '</pre>' );

    $show = mysql_numrows($end);

    $i=0;
while ($i < $show) {

      $user = mysql_result          ($end,$i,"user_name");
   $pass= mysql_result($end,$i,"password);

      echo '<pre>';
      echo 'ID: ' . $id . '<br>user
name: ' . $user . '<br>password: ' . $pass;
      echo '</pre>';

      $i++;
    }
  }
}
?>
```

Table-3 Comparison of three level of Coding:

S.no	SQLI query/string (conditions)	Low	Medium	High
1	**A' or' '='**	✓	✖	✖
2	**1' or 1=1**	✓	✖	✖
3	**1' or '1'='1**	✓	✖	✖
4	**union select null, @@ hostname#**	✓ **(Show host name of server)**	✖	✖
5	**A' or ''='**	✖	✓	✖
6	**1 or 1=1**	✖	✓	✖
7	**99 or 1=1 union select null,@@data dir**	✖	✓	✖
8	**%' or '0'='0**	✓	✖	✖
9	**1 or 1=1#**	✓**(Show admin)**	✓**(Show all record)**	✖

References:

[1] **Halfond, W. G., Jeremy Viegas, and Alessandro Orso. "A classification of SQL-injection attacks and countermeasures." Proceedings of the IEEE International Symposium on Secure Software Engineering. IEEE, 2006.**

[2] **Tajpour, Atefeh, et al. "SQL injection detection and prevention tools assessment." Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. Vol. 9. IEEE, 2010.**

[3] **Kindy, DialloAbdoulaye, and Al-Sakib Khan Pathan. "A Detailed Survey on Various Aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks, and Remedies." International Journal of Communication Networks and Information Security (IJCNIS) 5.2 (2013).**

[4] **Kumar, Puspendra, and R. K. Pateriya. "A Survey on SQL injection attacks, detection and prevention techniques." Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on. IEEE, 2012.**

[5] **Sadeghian, Amirmohammad, MazdakZamani, and AzizahAbdManaf. "A Taxonomy of SQL Injection Detection and Prevention Techniques." Informatics and Creative Multimedia (ICICM), 2013 International Conference on. IEEE, 2013.**

[6] **Elshazly, Khaled, et al. "A Survey of SQL Injection Attack Detection and Prevention." Journal of Computer and Communications 2014 (2014).**

[7] **Yeole, A. S., and B. B. Meshram. "Analysis of different technique for detection of SQL injection." Proceedings of the International Conference & Workshop on Emerging Trends in Technology. ACM, 2011.**

[8] **Ashish John, Ajay Agarwal, Manish Bhardwaj. An Adaptive Algorithm to Prevent SQL Injection. American Journal of Networks andCommunications. Special Issue: Ad Hoc Networks. Vol. 4, No. 3-1, 2015, pp. 12-15. doi: j.ajnc.s.2015040301.13**

[9] **Junjin, Mei. "An approach for SQL injection vulnerability detection." Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on. IEEE, 2009.**

[10] **http://www.computersecuritystudent.com/SECURITY_TOOLS/DVWA/DVWAv107/lesson1**