# SQL Injection 101

# Topics

- Introduction
- Attack Intent
- How SQL Injection works?
- Defence Against SQL Injection
- Other Injection Types
- SQL Injection tools
- Conclusion

# Introduction

- **SQL injection** is a code **injection** technique, used to attack data-driven applications, in which malicious **SQL** statements are inserted into an entry field for execution.

- This is a method to attack web applications that have a data repository.

- The attacker would send a specially crafted SQL statement that is designed to cause some malicious action.

# Attack Intent

- Determining **database** schema.
- Extracting **data**.
- Adding or modifying **data**.
- Bypassing **authentication**.

# How SQL injection works?

- The ability to inject **SQL commands** into the database engine through an existing application.
- SQL injection is the use of publicly available fields to gain entry to your database.
- This is done by entering SQL commands into your form fields instead of the expected data.
- Improperly coded forms will allow a hacker to use them as an entry point to your database.

# How SQL injection works?

1. App sends form to user.
2. Attacker submits form with SQL exploit data.
3. Application builds string with exploit data.
4. Application sends SQL query to DB.
5. DB executes query, including exploit, sends data back to application.
6. Application returns data to user.

# SQL Injection in PHP

```
$link = mysql_connect($DB_HOST, $DB_USERNAME,
$DB_PASSWORD) or  die ("Couldn't connect: " . mysql_error());

mysql_select_db($DB_DATABASE);

$query = "select count(*) from users where username =
'$username' and  password = '$password‘ ";

$result = mysql_query($query);
```

# SQL Injection attack on Login Forms

- `Attack:`
  - `password =` **’ or 1=1 --**
- SQL statement becomes:
  - **select count(*) from** *users* **where** *username = ‘user’* **and** *password =* **“ or 1=1 --**
  - Checks if password is empty OR 1=1, which is always true, permitting access.

# SQL Injection attack by appending 2nd Query

- Attack: foo'; **delete from table** *users* **where** *username* **like** '%admin%
- DB executes ***two*** SQL statements:
  - **select count(*) from** *users* **where** *username = 'user'* **and** *password= 'foo';* **delete from table** *users* **where** *username* **like** '%admin%'

# SQL Injection WAF Bypass

- Normal SQL Injection Basic Concent:
  - ?id=1’ union select 1,2,3,4,5--

- Bypassing WAF:
  - ?id=1’ /*!union*/ /*!select*/ 1,2,3,4,5--
  - ?id=1’/**/union/**/select/**/1,2,3,4,5--
  - ?id=1’+un/**/ion+se/**/lect+1,2,3--
  - ?id=1’+UNunionION+SEselectLECT+1,2,3--

# Defence againts SQL injection

- **1. Comprehensive data sanitization**
  - Web sites must filter / santize ***all*** user input.
  - For example, e-mail addresses should be filtered to allow only the characters allowed in an e-mail address.
- **2. Use a web application firewall**
  - A popular example is the free, open source module ModSecurity.
- **3. Limit database privileges by context**
  - Create multiple database user accounts with the minimum levels of privilege for their usage environment.
  - For example, the code behind a login page should query the database using an account limited only to the relevent credentials table.

# SQL Injection Tools

- BSQL Hacker
- SQLmap
- SQLninja
- Safe3 SQL Injector
- SQLSus
- Havij