**SQL injection**

1. **Brief**
2. **What is a Database?**
3. **What is SQL?**

SELECT \* {col\_name} FROM {table} WHERE {condition} LIMIT 2,1;

{condition} 🡪 (OR, AND), LIKE **‘**a%**’**

SELECT {col\_names} FROM {table1} **UNION** SELECT {col\_names} FROM {table2};

INSERT INTO {table} ({col\_name}**,** {col\_name}) VALUES (**'**{val\_1}**', '**{val\_2}**'**);

UPDATE {table} SET {col\_1} = **‘**{val\_1}**’,** {col\_2} = **‘**{val\_2}**’** WHERE {condition};

DELETE FROM {table} WHERE {condition};

DELETE FROM {table} 🡪 Empty the table

1. **What is SQL Injection?**

Using -- & ; as user input in SQL Statement

1. **In-Band SQLi:**

* In-Band SQL Injection
* Error-Based SQL Injection
* Union-Based SQL Injection

0 UNION SELECT 1,2,group\_concat(name, ':', password) FROM users

1. **Blind SQLi - Authentication Bypass:**

**' OR 1=1;--**

1. **Blind SQLi - Boolean Based:**

Trying Possibilities with LIKE operator until you found a match (return True ‘1’)

1. **Blind SQLi - Time Based:**

Trying Possibilities with LIKE operator until you found a match (**Delay**)

SLEEP(5) “4961”

1. **Out-of-Band SQLi:**
   1. An attacker makes a request to a website vulnerable to SQLI with an injection payload.
   2. The Website makes an SQL query to the database which also passes the hacker's payload.
   3. The payload contains a request which forces an HTTP request back to the hacker's machine
2. **Remediation:**

* Prepared Statements (With Parameterized Queries)
* Input Validation
* Escaping User Input