SQL INJECTION

By:

- Yashashwi Patil
- Mohammadsaad Mulla

Table of Contents

SQL REVISITED

- Structured Query Language.
- Language for managing relational databases.
- Databases include MySQL, PostgreSQL, Oracle, and Microsoft SQL Server.
- Commands include SELECT (retrieve data), INSERT (add data), UPDATE (modify data), and DELETE (remove data)

EmployID	FirstName	LastName	Salary
1	john	ryther	10000
2	Alex	Hamilton	20000
3	Sze	Chauhan	10000
4	Shiv	Chauhan	50000

Bagha, give me a list of customers who are from **अमरिका** !

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

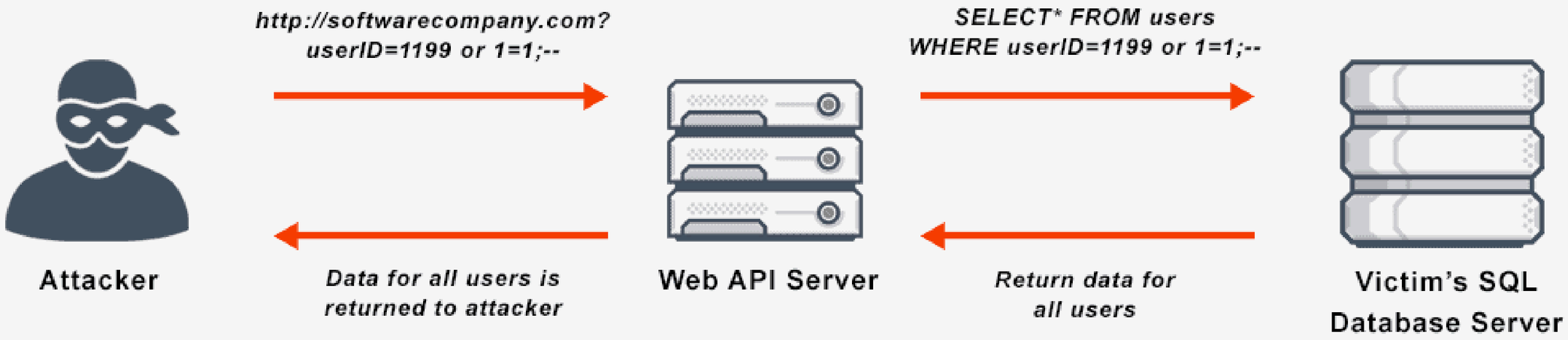
```
SELECT age, country
FROM Customers
WHERE country = 'USA';
```

age	country
31	USA
22	USA

SQL INJECTION

- SQL injection is a code injection technique used to attack databases.
- Manipulation of the SQL query to do the work it is not supposed to do.
- Most commonly used Web Penetration
- Tools like SQLMap & Acunetix are used for finding SQL Injections.
- SQL injection has been among top 5 vulnerabilities in OWASP Top 10 list.

HOW SQL INJECTION WORKS



DEMO TIME

https://www.hacksplaining.com/exercises/sql-injection

TYPES OF SQL INJECTION

Inband

- Attacker uses the same channel for attack and data retrieval.
- **Error-based**: Forces database errors to obtain information.
- **Union-based**: Fetches data from other table with the help of UNION operator.

Blind

- No data transfer, equally dangerous as in-band SQL injection.
- **Boolean-based:** Injects false queries to observe application responses.
- **Time-based**: Delays responses to detect successful queries.

Out-of-band

- Triggers external network connections, often via DNS or HTTP.
- Used when other techniques are not applicable.

SOME QUERIES

Error-Based SQLi:

www.random.com/app.php?id=’

You have an error in your SQL syntax

Union-Based SQLi:

www.random.com/app.php?id=’ UNION select username, password from users--

select ? from table1 union NULL

SOME QUERIES

Boolean-Based SQLi:

- www.random.com/app.php?id=1 and SUBSTRING((select Password from users where Username = ‘Administrator’), 1, 1) = ‘s’

  No result → s is not the first letter of password

- www.random.com/app.php?id=1 and SUBSTRING((select Password from users where Username = ‘Administrator’), 1, 1) = ‘e’

  Result obtained → e is the first letter of password

PREVENTION

1) Use Parameterized Statements (Prepared Statements): Use placeholders to separate SQL code from user input, preventing injection.

```
const prn = 22620012; // User-provided input

const sql = 'SELECT name FROM students WHERE prn = ?';

connection.query(sql, [prn], (err, results) => {
  if (err) {
    console.error('Error executing query: ' + err.message);
    return;
  }
});
```

PREVENTION

2) Input Validation and Sanitization: Validate and sanitize user inputs to ensure they match expected formats and are safe to use in SQL queries.

```
user_id = input("Enter User ID: ")
if not user_id.isdigit():
    # Handle invalid input
else:
    user_id = int(user_id)
    cursor.execute("SELECT username FROM users
WHERE id = ?", (user_id,))
```

PREVENTION

3) **Least Privilege Principle:** Limit database user permissions to the minimum necessary for application functionality to reduce the attack surface.

4) **Web Application Firewall** (**WAF**)**:** Deploy a WAF to filter and block malicious SQL injection attempts before they reach your application.

- https://demo.testfire.net/login.jsp


THANKS FOR OUR ATTENTION GUYS
GREAT AUDIENCE!
makeameme.org