

# Problem Set 2

## What are Injection Attacks and how does it work?

Injection attack is a form of injecting code through a vulnerability that is executed by the application.

This happens with untrusted or poor handled data, as allowed characters, data format and amount of expected data.

## Why is it important to prevent Injection Attacks?

They could cause data loss, corruption (Integrity in CIA triad), disclosure to unauthorized parties (Confidentiality) or denial of access (Availability)

## Try It! String SQL Injection

### Try It! String SQL Injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating strings making it susceptible to String SQL injection:

```
"select * from user_data where LAST_NAME = '" + userName + "'";
```

Using the form below try to retrieve all the users from the users table. You shouldn't need to know any specific user name to get the complete list, however you can use 'Smith' to see the data for one user.

☒

Account Name:

You have succeeded!

| USERID | FIRST  | LAST                 | NUMBER        | CC_TYPE | COOKIE | LOGIN_COUNT |
|--------|--------|----------------------|---------------|---------|--------|-------------|
| 101    | Joe    | Snow                 | 987654321     | VISA    | ,      | 0           |
| 101    | Joe    | Snow                 | 2234200065411 | MC      | ,      | 0           |
| 102    | John   | Smith                | 2435600002222 | MC      | ,      | 0           |
| 102    | John   | Smith                | 4352209902222 | AMEX    | ,      | 0           |
| 103    | Jane   | Plane                | 123456789     | MC      | ,      | 0           |
| 103    | Jane   | Plane                | 333498703333  | AMEX    | ,      | 0           |
| 10312  | Jolly  | Hershey              | 176896789     | MC      | ,      | 0           |
| 10312  | Jolly  | Hershey              | 333300003333  | AMEX    | ,      | 0           |
| 10323  | Grumpy | youaretheweakestlink | 673834489     | MC      | ,      | 0           |
| 10323  | Grumpy | youaretheweakestlink | 33413003333   | AMEX    | ,      | 0           |
| 15603  | Peter  | Sand                 | 123609789     | MC      | ,      | 0           |
| 15603  | Peter  | Sand                 | 338893453333  | AMEX    | ,      | 0           |
| 15613  | Joesph | Something            | 33843453533   | AMEX    | ,      | 0           |
| 15837  | Chaos  | Monkey               | 32849386533   | CM      | ,      | 0           |
| 19204  | Mr     | Goat                 | 33812953533   | VISA    | ,      | 0           |

We can inject when we escape the input and select an always true condition, so we select \* from table

### Before patch

```
protected void InjectableQuery(String accountName) {
    try {
        Connection connection = DatabaseUtilities.getConnection(getWebSession());
        String query = "SELECT * FROM user_data WHERE last_name = '" + accountName + "'";

        try {
            Statement statement = connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
            ResultSet results = statement.executeQuery(query);
        }
    }
}
```

### After Patch

```
1
64 try {
1   PreparedStatement pstmt = connection.prepareStatement(query, ResultSet.TYPE_SCROLL_IN
2       ResultSet.CONCUR_READ_ONLY);
3
4   pstmt.setString(1, accountName);
5   ResultSet results = pstmt.executeQuery(query);
6
```