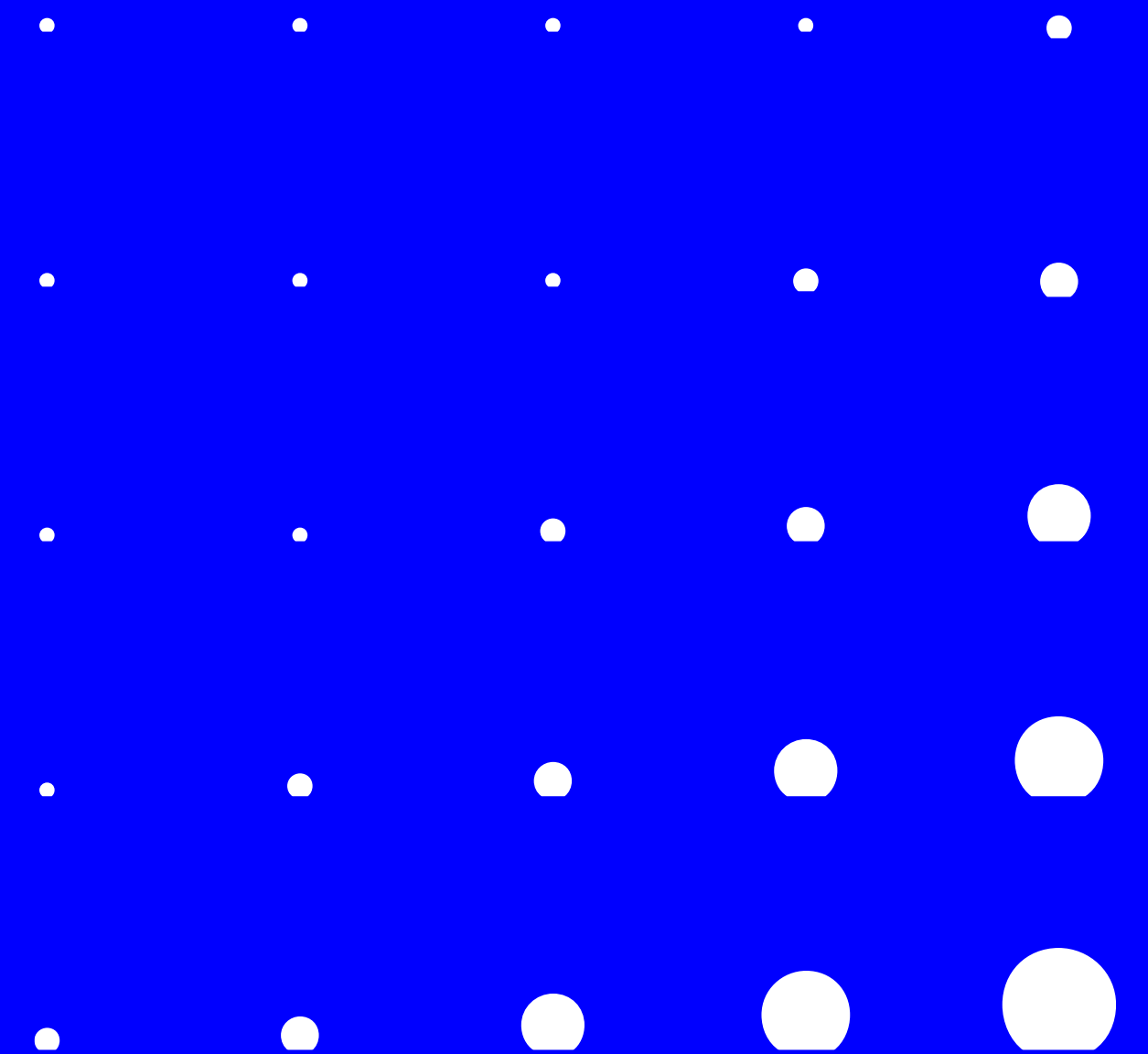


## SQL-Injections und Prepared Statements



# Was ist eine SQL-Injection?

- Eine SQL-Injection bezeichnet das Ausnutzen einer Sicherheitslücke in Zusammenhang mit SQL-Datenbanken, die durch mangelnde Maskierung oder Überprüfung von Metazeichen in Benutzereingaben entsteht.
  - SQL-Injections sind dann möglich, wenn Daten wie beispielsweise Benutzereingaben in den SQL-Interpreter gelangen.
  - Denn Benutzereingaben können Zeichen enthalten, die für den SQL-Interpreter Sonderfunktion besitzen und so Einfluss von außen auf die ausgeführten Datenbankbefehle ermöglichen.
  - Solche Metazeichen in SQL sind zum Beispiel
    - der Backslash,
    - das Anführungszeichen,
    - der Apostroph und
    - das Semikolon.

# Was ist eine SQL-Injection?

- Der Angreifer versucht dabei, über die Anwendung, die den Zugriff auf die Datenbank bereitstellt, eigene Datenbankbefehle einzuschleusen.
- Sein Ziel ist es,
  - Daten auszuspähen,
  - Daten in seinem Sinne zu verändern,
  - die Kontrolle über den Server zu erhalten oder
  - einfach größtmöglichen Schaden anzurichten.

# SQL-Injection: Veränderung von Daten

	Erwarteter Aufruf
Aufruf	<code>http://webserver/cgi-bin/find.cgi?ID=42</code>
Erzeugtes SQL	<code>SELECT author, subject, text FROM artikel WHERE ID=42;</code>
	SQL-Injection
Aufruf	<code>http://webserver/cgi-bin/find.cgi?</code> <code>ID=42;UPDATE+USER+SET+TYPE="admin"+WHERE+ID=23</code>
Erzeugtes SQL	<code>SELECT author, subject, text FROM artikel WHERE</code> <code>ID=42;UPDATE USER SET TYPE="admin" WHERE ID=23;</code>

# SQL-Injection: Datenbank-Server verändern

	Erwarteter Aufruf
Aufruf	http://webserver/search.aspx?keyword=sql
Erzeugtes SQL	SELECT url, title FROM myindex WHERE keyword LIKE '%sql%'
	SQL-Injection
Aufruf	http://webserver/search.aspx? keyword=sql '+;GO+EXEC+cmdshell('shutdown+/s')+--
Erzeugtes SQL	SELECT url, title FROM myindex WHERE keyword LIKE '%sql ' ;GO EXEC cmdshell('shutdown /s') --%'

# SQL-Injection: Ausspähen von Daten

	Erwarteter Aufruf
Aufruf	http://webserver/cgi-bin/find.cgi?ID=42
Erzeugtes SQL	SELECT author, subject, text FROM artikel WHERE ID=42;
	SQL-Injection
Aufruf	http://webserver/cgi-bin/find.cgi? ID=42+UNION+SELECT+login,+password,+ 'x'+FROM+user
Erzeugtes SQL	SELECT author, subject, text FROM artikel WHERE ID=42 UNION SELECT login, password, 'x' FROM user;

# SQL-Injection: generelle Schwachstelle

```
String benutzereingabe = ...;
```

```
...
```

```
Statement stmt      = conn.createStatement();
```

```
ResultSet rs        = stmt.executeQuery(
```

```
    "SELECT spalte1 FROM tabelle WHERE "+
```

```
    "spalte2 = ' " + benutzereingabe + " ' ;") ;
```

# Prepared Statements



# Was sind Prepared-Statements?

- Ein Prepared Statement ist eine sogenannte vorbereitete Anweisung für ein DBS.
- Im Gegensatz zu gewöhnlichen Statements enthält es noch keine Parameterwerte.
- Stattdessen werden dem DBS Platzhalter übergeben.
- Mittels Prepared Statements können SQL-Injections effektiv verhindert werden, da das DBS die Gültigkeit von Parametern prüft, bevor diese verarbeitet werden.
- Soll ein Statement mit unterschiedlichen Parametern mehrere Male (z. B. innerhalb einer Schleife) auf dem Datenbanksystem ausgeführt werden, können Prepared Statements zusätzlich einen Geschwindigkeitsvorteil bringen, da das Statement schon vorübersetzt im DBS vorliegt und nur noch mit den neuen Parametern ausgeführt werden muss.

# Der anfällige Code: Jeder Code mit äußeren Parametern!

```
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class Schreiben_01 {

    public static boolean schreibe_Produkt(
        String name, String beschreibung, BigDecimal preis, int lagerbestand) {
        // Datenbank-Parameter:
        String url = "jdbc:mysql://localhost:3306/dm"; // IP, TCP-Port und Datenbankname
        String user = "root"; // MySQL-Benutzername
        String password = ""; // Passwort dieses Benutzers
        // Verbindung herstellen:
        try {
            Connection conn = DriverManager.getConnection(url, user, password);
            Statement stmt = conn.createStatement();
            // SQL-Select absetzen:
            String sql = "INSERT INTO produkt (name,beschreibung,standardpreis,lagerbestand) ";
            sql += "VALUES (";
            sql += "'" + name + "', ";
            sql += "'" + beschreibung + "', ";
            sql += preis + ", ";
            sql += lagerbestand + ")";
            stmt.execute(sql);
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public static void main(String[] args) {
        boolean erfolg = schreibe_Produkt("Test-Name","Test-Beschreibung",new BigDecimal(200.99),100);
        System.out.println("Erfolg: " + erfolg);
    }
}
```

...bei jedem Parameter,  
der von außen kommt!

# Der sichere Code: Verwendung von Prepared Statements!

```
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Schreiben_02 {
    public static boolean schreibe_Produkt(
        String name, String beschreibung, BigDecimal preis, int lagerbestand) {
        // Datenbank-Parameter:
        String url = "jdbc:mysql://localhost:3306/dm"; // IP, TCP-Port und Datenbankname
        String user = "root"; // MySQL-Benutzername
        String password = ""; // Passwort dieses Benutzers
        // Verbindung herstellen:
        try {
            Connection conn = DriverManager.getConnection(url, user, password);
            // SQL-Select vorbereiten:
            String sql = "INSERT INTO produkt (name,beschreibung,standardpreis,lagerbestand) ";
            sql += "VALUES (?, ?, ?, ?)";
            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setString(1, name);
            ps.setString(2, beschreibung);
            ps.setBigDecimal(3, preis);
            ps.setInt(4, lagerbestand);
            // SQL-Select absetzen:
            ps.execute();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public static void main(String[] args) {
        boolean erfolg = schreibe_Produkt("Test-Name", "Test-Beschreibung", new BigDecimal(200.99), 100);
        System.out.println("Erfolg: " + erfolg);
    }
}
```

Ein ? nach dem anderen besetzen...  
Alle müssen besetzt werden!