Angriffe auf Datenhaltung

SQL Injections

Daniel Schosser

Chair for Network Architectures and Services
Institut für Informatik
Technische Universität München

16. Dezember 2011

Agenda

1. Einführung
   - Was ist SQL und wie funktioniert es?
   - Was ist eine SQL Injections?
2. Möglichkeiten für einen Angriff
   - Wo kann ein Angriff ansetzen?
   - Mögliche Schäden?
3. Lösungen & Ursachen
4. Zusammenfassung

Abbildung: SQL Injection Versuch

Agenda

Was ist SQL?

- **S**tructured **Q**uery **L**anguage
- Basiert auf SEQUEL (Structured English Query Language) von Edgar F. Codd (IBM) aus den 1970er Jahren
- SQL ist eine Datenbanksprache zur Kommunikation zwischen einer Anwendung und einer relationalen Datenbank
- SQL1 wurde 1986 erstmals Standard
- SQL2 löst 1992 den Vorgänger als Standard ab
- SQL3 (SQL:1999) wird 1999 als Standard verabschiedet
- SQL:2003, SQL:2006, SQL:2008

Was ist SQL?

- Drei verschiedene Aufgabentypen sind per Statement möglich

1. Rechteverwaltung
2. Definition des Datenbankschemas
3. Datenmanipulation (Ändern, Einfügen, Löschen)

Wie ist ein Statement aufgebaut?

- Beispiel:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

- Ein Statement besteht aus drei Teilen
  1. SQL-Syntax
  2. Daten
  3. Logikteil

- **Metazeichen (*, %, ', - -, #, ...)**

Wo wird SQL verwendet?

Verwendung:

- Webseiten
- Soziale Netzwerke
- Spiele
- Smartphones
- Firmennetzwerke
- Behörden

Agenda

Was ist eine SQL-Injection?

- Wie wird eine SQL-Injection ermöglicht?
  - Ausnutzung von Sicherheitslücken in Anwendungen, welche Zugriff auf eine Datenbank haben
- Warum ist das gefährlich?
  - Einschleusen oder ändern von Befehlen in Datenbanken

Was ist eine SQL-Injection?

Abbildung: Loginformular einer Webseite

Was ist eine SQL-Injection?

Beispiel:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

Was ist eine SQL-Injection?

Beispiel:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

1. Angreifer weiß, dass es einen Benutzer mit dem Namen "John" gibt.

Was ist eine SQL-Injection?

Beispiel:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

1. Angreifer weiß, dass es einen Benutzer mit dem Namen ”John” gibt.
2. Angreifer gibt als Benutzernamen ”John;’ - -” ein und lässt das Passwortfeld leer

Was ist eine SQL-Injection?

Beispiel:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

1. Angreifer weiß, dass es einen Benutzer mit dem Namen "John" gibt.
2. Angreifer gibt als Benutzernamen "John;' - -" ein und lässt das Passwortfeld leer

```
SELECT *
FROM users
WHERE name='John;' - -' AND password='_______';- -
```

Agenda

1 Einführung
- Was ist SQL und wie funktioniert es?
- Was ist eine SQL Injections?

2 Möglichkeiten für einen Angriff
- Wo kann ein Angriff ansetzen?
- Mögliche Schäden?

3 Lösungen & Ursachen

4 Zusammenfassung

Wo kann ein Angriff ansetzen?

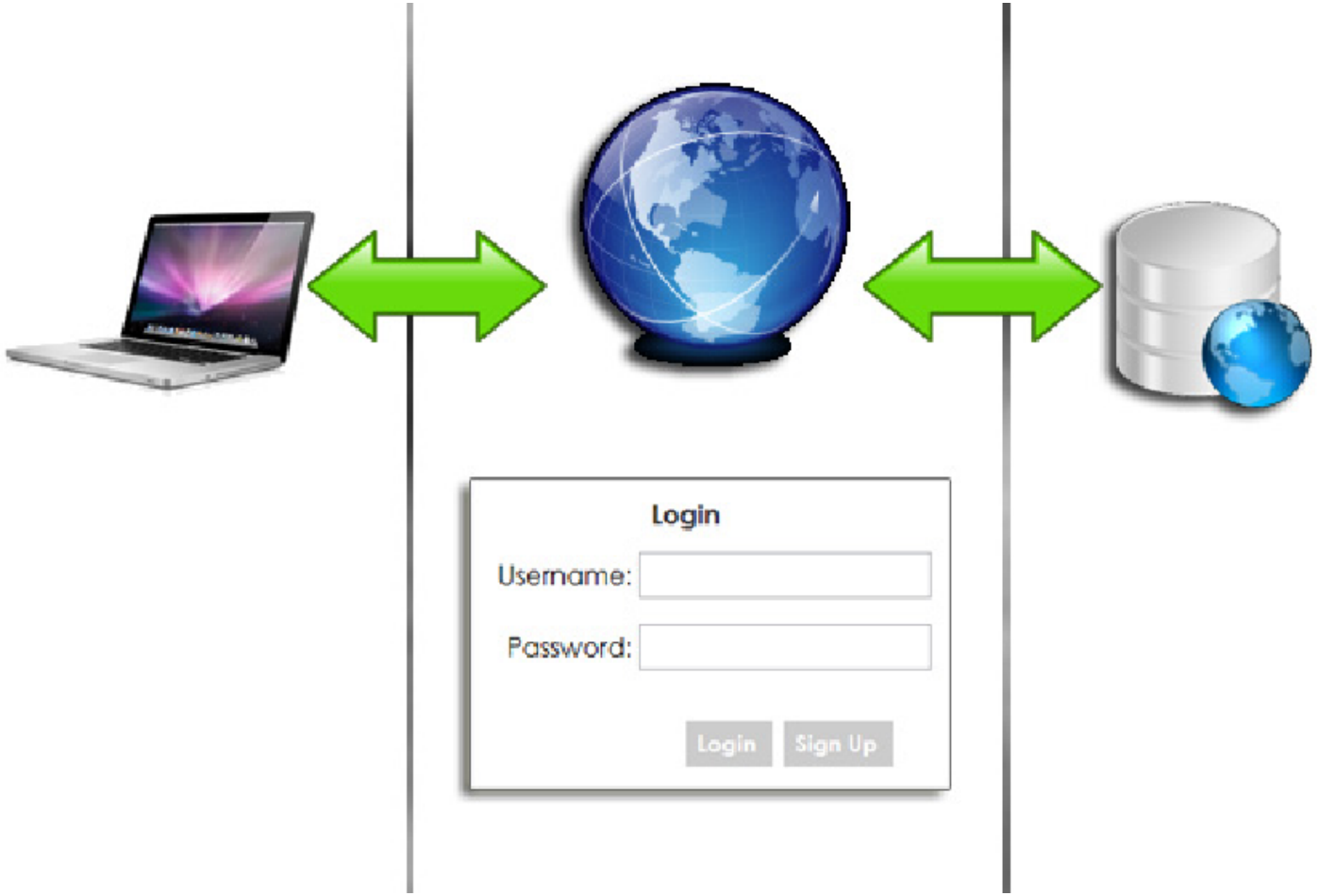

Abbildung: Kommunikation zwischen Nutzer und Datenbank

Mögliche Schäden? Daten hinzufügen

- Neue Benutzer hinzufügen um Zugang zu gesperrten Bereichen zu erhalten
- Artikel auf fremden Seiten veröffentlichen
- ...

Mögliche Schäden? Daten hinzufügen

Abfrage:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

Mögliche Schäden? Daten hinzufügen

Abfrage:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

- Passwort:
  abc'; INSERT INTO users (name, passwort, email) VALUES ('anonymous', 'geheim', 'anonymous@example.com')

Mögliche Schäden? Daten hinzufügen

Neue Abfrage:

```
SELECT *
FROM users
WHERE name='nichtWichtig' AND password='abc';
INSERT INTO users (name, passwort, email)
VALUES ('anonymous', 'geheim', 'anonymous@example.com')';- -
```

Mögliche Schäden? Autorisierung

- Passwortabfrage auskommentieren
- SQL Anfrage durch logische Operatoren erweitern
- (Neuen Benutzer hinzufügen)
- ...

Mögliche Schäden? Autorisierung

Abfrage:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

Mögliche Schäden? Autorisierung

Abfrage:

```
SELECT *
FROM users
WHERE name='_______' AND password='_______';- -
```

- Nutzername: ”John”
- Passwort: *abc’ OR WHERE 1=1*

Mögliche Schäden? Autorisierung

Neue Abfrage:

```
SELECT *
FROM users
WHERE name='John' AND password='abc' OR WHERE 1=1';- -
```

Mögliche Schäden? Daten verändern

- Daten eines Nutzers abändern
- Eigenen Quellcode in Artikel auf fremden Seiten einbauen (Iframe, Javascript, ...)
- Einstellungen in gesperrten Bereichen ändern
- ...

Beispiel:

```
UPDATE users SET password='hallo' WHERE 1=1
```

Mögliche Schäden? Daten löschen

- Komplette Datenbank leeren
- Alle Benutzer löschen und somit den Zugang sperren
- ...

Beispiel:

- **DELETE FROM** users **WHERE** level='professor'; - -
- **GO EXEC** cmdshell('shutdown /s'); - -
- Robert'); **DROP TABLE** Students; - -

Mögliche Schäden? Daten sammeln

- Sammeln von Nutzerinformationen
- Benutzernamen, Passwörter, Emailadressen, . . .
- Ausgabe der Daten ist hierbei das Hauptproblem, da die Anwendung ein bestimmtes Ergebnis erwartet und ein anderes bekommt
  - UNION nutzen und Spalten vermischen
  - Fehlermeldungen erzwingen

Beispiel:

- **SELECT** * **FROM** users **WHERE** 1=1

Agenda

1 Einführung
- Was ist SQL und wie funktioniert es?
- Was ist eine SQL Injections?

2 Möglichkeiten für einen Angriff
- Wo kann ein Angriff ansetzen?
- Mögliche Schäden?

3 Lösungen & Ursachen

4 Zusammenfassung

Lösungen - Eingaben Validierung

- Ein Feld darf nur Buchstaben, nur Zahlen, ... enthalten
- Eingabelänge beschränken
- Entweder direkt bei der Formulareingabe nur spezielle Zeichen zulassen
- oder vor dem Abschicken an die Datenbank umwandeln, z.B. (int) $age

Lösungen - Eingaben Validierung

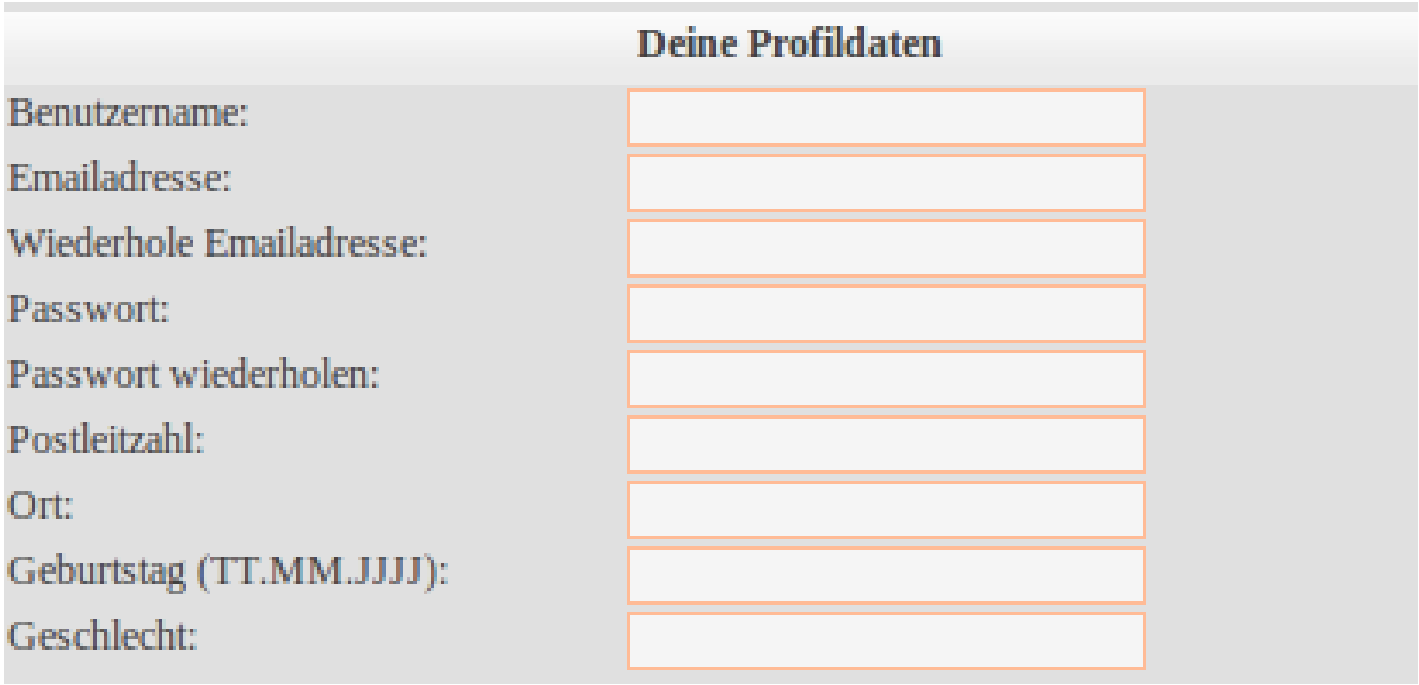

Abbildung: Registrierungsformular

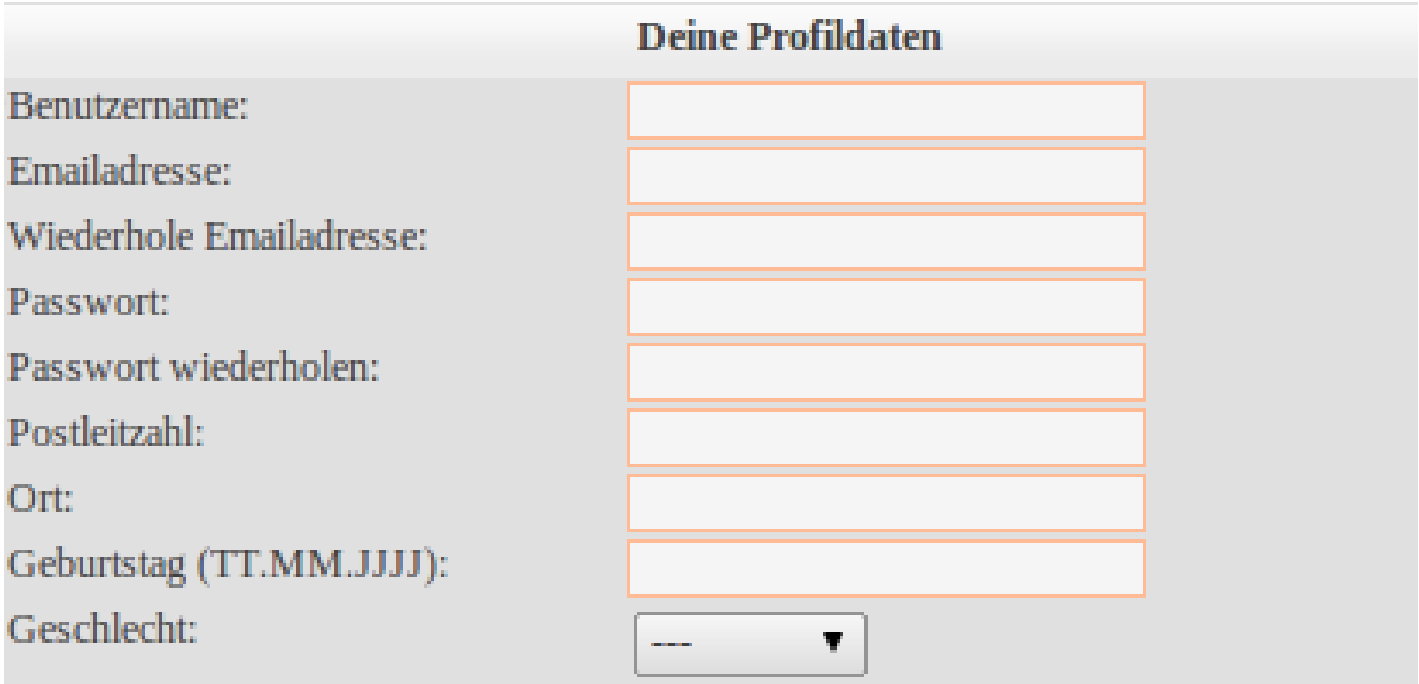

Abbildung: Registrierungsformular mit Auswahlmöglichkeit

Lösungen - Eingaben Validierung

- ALLE Daten überprüfen
- Häufig vergessen: Selectboxen
- Nur weil ein Nutzer eine Option aus einer Liste auswählen darf heißt das nicht, dass auch ein korrektes Element abgeschickt wird

Lösungen - Schlüsselwörter abfangen

- Alle Nutzereingaben vor dem Absenden auf Schlüsselwörter (SELECT, INSERT, DELETE, UPDATE, ...) überprüfen
- $\Rightarrow$ bei einem englischen Freitext können die Wörter trotzdem vorkommen
- Entspricht der fertig zusammengesetze Query der gewünschten Abfrage? (Userabfrage enthält nur SELECT und kein DELETE, UPDATE oder INSERT)
- Kommen Schlüsselwörter oder Metazeichen in der Anfrage vor, die das Ergebnis verfälschen könnten?

Lösungen - Escapen

- Eingaben von Nutzern sollten grundsätzlich ”escaped” werden
- Alle Metazeichen werden durch einen vorrangehenden ”Backslash” markiert

Beispiel:
Ein Nutzer schreibt in seine Profilbeschreibung: *”Mir geht’s gut”* und speichert.

- Das SQL-Statement würde durch das ’-Zeichen davon ausgehen, dass es sich um zwei seperate Parameter handelt.
- Durch das Escapen weiß der Interpreter, dass der String nur ein Parameter ist

Lösungen - Prepared Statements

- Die Formulardaten werden an das Statement oder eine Funktion übergeben
- Alle Daten werden validiert
- Die Daten werden an den entsprechenden Stellen in das Statement eingefügt

```php
public function fetchAllByUserId( $uid )
{
    return $this->fetchAllAbstract(
        $this->getBaseSelect()
             ->where( $this->_tableShort.'.user_id = ?', $uid )
    );
}
```

Abbildung: Prepared Statement in OOP

Lösungen - Rechteverwaltung

- Der Anwendung nur die benötigten Rechte geben
- Mehrere Nutzer anlegen für verschiede Aufgaben

Agenda

1 Einführung
- Was ist SQL und wie funktioniert es?
- Was ist eine SQL Injections?

2 Möglichkeiten für einen Angriff
- Wo kann ein Angriff ansetzen?
- Mögliche Schäden?

3 Lösungen & Ursachen

4 Zusammenfassung

All input is evil, until proven otherwise!

Zusammenfassung

- Egal wie klein oder unwichtig eine Nutzereingabe auch erscheinen mag muss der Inhalt auf Korrektheit/Gültigkeit geprüft werden

Zusammenfassung

- Egal wie klein oder unwichtig eine Nutzereingabe auch erscheinen mag muss der Inhalt auf Korrektheit/Gültigkeit geprüft werden
- Angreifer können: Daten löschen, ändern, hinzufügen, sammeln, ...

Zusammenfassung

- Egal wie klein oder unwichtig eine Nutzereingabe auch erscheinen mag muss der Inhalt auf Korrektheit/Gültigkeit geprüft werden
- Angreifer können: Daten löschen, ändern, hinzufügen, sammeln, ...
- Schutzmechanismen: Regular Expressions, Casten, Escapen, Prepared Statements ...

Fragen?

Vielen Dank für die Aufmerksamkeit!