itEXPERsT
Your Data Guard

Dipl. Ing. Christian Perst
Ger. zert. Sachverständiger, IT-Forensiker
IT-Security-Consultant

www.itEXPERsT.at

SQL-Injection erklärt

Was ist eigentlich SQL-Injection (Abkürzung SQLi)?

SQL-Injection ist eine Angriffsart auf IT-Systeme, bei der über eine Webanwendung Befehle für eine Datenbank eingeschleust werden. Dadurch ist es z.B. möglich, Benutzer anzulegen, Passwörter auszulesen, Daten zu manipulieren oder ganzen Datenbanken zu löschen.

Oft ist eine Webanwendung, wie z.B. ein Webshop, an eine Datenbank angebunden. Um mit der Datenbank Daten auszutauschen, wird häufig die Sprache SQL verwendet. SQL steht für Standard Query Language (Standardabfragesprache) und wird von vielen Datenbanken verstanden.

Über Befehle werden Werte ausgelesen oder bearbeitet und die Datenstruktur verändert.

Wie muss ich mir SQL vorstellen?

Als Beispiel stelle man sich eine einfache Tabelle mit dem Namen user in einer Datenbank vor. In dieser Tabelle könnten Adressen gespeichert werden.

Für eine bessere Übersicht, sind die Befehle der SQL-Abfrage in blau, die Variablen wie Tabellen oder Spaltennamen in schwarz und die Eingaben in grün gehalten.
Ein paar häufige Abfragen:

Anzeige aller Werte in der Tabelle user
```
SELECT * FROM user;
```

Anzeige aller Werte in der Tabelle user, deren Name itEXPERsT ist
```
SELECT * FROM user WHERE name='itEXPERsT';
```

Anzeige von einigen Werten (hier strasse, plz und ort) der Tabelle user, deren Name itEXPERsT ist
```
SELECT strasse, plz, ort FROM user WHERE name='itEXPERsT';
```

EXPERsT your IT

Anzeige von allen Werten der Tabelle user, deren Name itEXPERsT ist und deren Passwort penetrationstest ist. (Anmerkung: Dies ist ein schwaches Passwort und sollte nicht in der Praxis verwendet werden.) Eine Abfrage in dieser Art wird oft verwendet, um einen Login in eine Webplattform zu ermöglichen.

```
SELECT * FROM user WHERE name='itEXPERsT' AND password='penetrationstest';
```

Ändere das Passwort des Datensatzes mit dem Namen itEXPERsT in it-forensik

```
UDATE user SET password='it-forensik' WHERE name='itEXPERsT';
```

Lösche die Tabelle user

```
DROP TABLE user;
```

Lösche die Datenbank meine_gesamte_datenbank

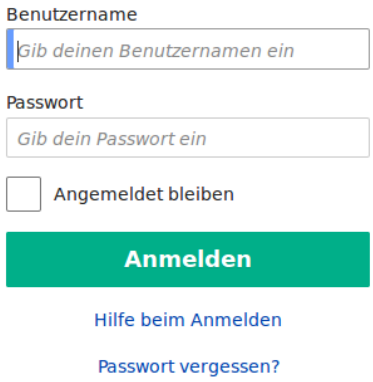
```
DROP DATABASE meine_gesamte_datenbank;
```

Einige besondere Zeichen seien noch erwähnt.

Zeichen	Bedeutung
;	Das Semikolon ist das Zeichen für das Befehlende.
--	Alle Zeichen nach dem Kommentar (zwei Minuszeichen) werden ignoriert.
'	Das einfache Hochkomma begrenzt eine Zeichenkette, wie z.B. 'itEXPERsT' als Benutzername.

Wie funktioniert nun SQL-Injection?

Angenommen, wir wollten einen Login-Versuch manipulieren. Dazu gibt es ein Anmeldefenster wie dieses hier. Der Benutzer meldet sich mit Namen und Passwort an.



Das Programm, das auf dem Server läuft, führt vereinfacht folgende Abfrage aus.

```
Schleife
        Wenn    (Abfrage_liefert_Wert)
        dann
                login_mit_name <name>
        sonst
                Fehlerausgabe: Benutzername oder Passwort falsch.
Schleife_Ende
```

Im Hintergrund steht eine SQL-Abfrage folgender Art.

```
SELECT * FROM user WHERE name='$name' AND password='$passwort';
```

Das einfache Hochkomma begrenzt eine Zeichenkette. Der Inhalt der Zeichenkette besteht aus dem Wert, der für die Abfrage verwendet wird. Der Wert $name steht für eine Variable, und zwar für den Inhalt, den der Benutzer im Feld Benutzername eingibt. Genauso steht $passwort' für den Inhalt im Feld Passwort.

Wird der richtige Name mit dem richtigen Passwort eingegeben, werden die Daten des Nutzers als Antwort auf die Frage zurückgeliefert. Gibt nun die Select-Abfrage Werte zurück, wird das Login durchgeführt. Gibt die Abfrage keine Werte aus, ist der Name, das Passwort oder beides falsch. Es erfolgt kein Login.

Das Ziel eines erfolgreichen Angriffs ist also, dass ein Datensatz eines Benutzers ausgegeben wird. Trifft die Anwendung keinen Schutz gegen SQL-Injection, können beliebig komplexe Abfragen in das Feld Benutzername eingegeben werden.

Ein Benutzer kann z.B. ein einfaches Hochkomma (') bei Benutzername angeben. Die SQL-Abfrage führt zu

```
SELECT * FROM user WHERE name=''' AND password='$passwort';
```

Bitte beachten Sie die drei Hochkomma. Damit gibt es zwei einfache Hochkommas vor dem Benutzernamen, der in dem Fall leer ist, und ein einfaches Hochkomma danach. Der SQL-Standard legt jedoch fest, dass eine Zeichenkette nur mit einem Hochkomma vorne und hinten begrenzt wird. Das ergibt einen Fehler und eine Fehlermeldung wird angezeigt. Damit haben wir einen Hinweis, dass SQL-Injection möglich ist.

Man kann nun z.B. folgende Zeichenketten bei Benutzername admin und bei Passwort ' OR 1=1; – eingeben. Die Abfrage führt zu

```
SELECT * FROM user WHERE name='admin' AND password='' OR 1=1; –';
```

Damit ist der Benutzername admin und das Passwort leer. Jedoch wurde noch ein Vergleich angefügt, der immer wahr ist (1=1). Die Abfrage mit dem leeren Passwort liefert kein Ergebnis. In der Logik ist das Ergebnis ein false, also falsch oder 0.

```
SELECT * FROM user WHERE name='admin' AND password='' OR 1=1; –';
                                                  (false) OR 1=1; –';
```

Der immer wahre Vergleich (1=1) liefert in der Abfrage ein logisches true, wahr oder auch 1.

```
                                                  (false) OR (true); –';
```

Nachdem dieses Wahr mit dem Falsch mit einem logischen Oder verknüpft ist, ergibt sich

```
                                                          (true); –';
```

und das ist 1. Der Kommentar verwirft alles, was dahinter steht, und beendet somit unsere SQL-Abfrage.

Also wird etwas, was nicht leer ist, an das Programm zurückgeliefert. Ein nicht leerer Wert bedeutet für das Programm, es kann die Login-Prozedur mit dem Benutzernamen starten. Voila - wir sind als Benutzername admin eingeloggt.

Wie kann man SQL-Injection verhindern?

Der Manipulation von SQL-Injection muss gleich bei der Entwicklung ein Riegel vorgeschoben werden. Ein Prinzip in der Softwareentwicklung heißt „all input is evil“, also alle Eingaben und Daten, die vom Benutzer kommen, müssen als fehlerhaft und bösartig (als Manipulationsversuch) angesehen werden.

Ein Weg, um SQL-Injections zu verhindern, besteht darin, nur solche Zeichen in der Eingabe zuzulassen, die erlaubt sind. Benutzer dürfen somit keine Hochkommas und SQL-spezifischen Zeichen mehr in Namen, Passwörtern, Adressen, Artikelnummern verwenden. Dabei handelt es sich um eine Projekt-Anforderung.

Leider ist es über verschiedene Darstellungen von Zeichen in anderen Zeichensätzen, z.B. Unicode, möglich, diese Filterung zu umgehen. So ist das Hochkomma auch als U+0027 darstellbar. Damit wird das Ausfiltern von SQL-spezifischen Zeichen schwierig. Eine Alternative wäre, nur die gültigen Zeichen zuzulassen, also das deutsche Alphabet und einige unkritische Sonderzeichen für diverse Umlaute. Damit würde man wieder zu den 256-Zeichen des ASCII-Zeichensatzes und einigen fest definierten Zeichen aus bestimmten Zeichensätzen zurückkehren. Ob dies eine Option ist, muss das Entwicklungsteam entscheiden.

Eine bessere Lösung besteht meist darin, sogenannte Stored-Procedures (gespeicherte Prozeduren) einzusetzen. Hierbei wird die Abfrage abstrahiert gespeichert. Diese Stored-Procedures sind frei von Manipulationsversuchen und eine SQL-Injection kann damit nicht mehr durchgeführt werden.

Über den Autor

Dipl. Ing. Christian Perst ist Hackerabwehrer und Datendetektiv. Er ist Experte für IT-Penetrationstest und IT-Forensik (www.itEXPERsT.at). Er arbeitet auch als gerichtlich beeideter Sachverständiger für forensische Datensicherung, Datenrekonstruktion, Datenauswertung.
Er hat SANS-Zertifizierungen als Incident Handler und Forensic Analyst.