E Ergänzende Erklärungen zu Webapplikationen

Im Folgenden werden für Webapplikationen typische Sicherheitslücken sowie das für das Verständnis notwendige Basiswissen aufgeführt.

E.1 SQL Injection

Bei der Structured Query Language (SQL) handelt es sich um eine Abfragesprache für relationale Datenbanken. Diese Datenbanken übernehmen in vielen Webapplikationen die Aufgabe der Speicherung von Informationen. Aus dieser Tatsache heraus ergibt sich die Möglichkeit für die Anwender, Informationen aus der Datenbank abzufragen. Abfragen werden bei der internen Verarbeitung in Form von SQL-Kommandos formuliert. In diese Abfragen fließen häufig Benutzereingaben mit ein. Ein Beispiel für eine solche Abfrage stellt eine Suchanfrage dar, bei der die gesuchte Zeichenkette als Abfragewert in das SQL-Kommando übernommen wird. Erfolgt keine ausreichende Prüfung der eingegebenen Werte, kann es einem Angreifer gelingen, das SQL-Kommando abzuändern und eventuell sogar beliebige weitere SQL-Kommandos zur Ausführung zu bringen. Dieser Vorgang wird als SQL Injection bezeichnet.

Beispiel Webforum

Ein Webforum speichert die Benutzerinformationen in einer Tabelle mit dem Namen `User` in der Datenbank des Forums. Die Tabelle enthält unter anderem die Spalten `Benutzer-ID`, `Name`, `RealName`, `Status`, `Ort` und `Passwort`. Der Status „normal" steht für einfache Anwender, der Status „admin" verleiht dem Anwender Administratorenrechte.

Der folgende Befehl listet alle Anwender aus Berlin auf, sowie alle anderen Informationen, die zu diesen Anwendern in der Tabelle `User` vermerkt sind:

```
SELECT * FROM User WHERE Ort = 'Berlin'
```

Ein Angreifer könnte hier versuchen, zusätzlichen Code einzufügen. Die Eingabe der Zeichenkette `Berlin'; DROP TABLE User --` könnte dann den folgenden SQL-Befehl ergeben:

```
SELECT * FROM User WHERE Ort = 'Berlin'; DROP TABLE User --'
```

Das Semikolon dient bei SQL als Trennzeichen für Kommandos. Zunächst wird somit der `SELECT`-Befehl ausgeführt und danach die Tabelle `User` gelöscht, falls der Datenbankbenutzer die nötigen Rechte besitzt. Die Zeichenkette „`--`" sorgt dafür, dass die letzten einfachen Anführungszeichen als Kommentar angesehen und somit ignoriert werden und keinen Fehler verursachen.

Eine weitere Eingabemöglichkeit ist folgende:

```
SELECT * FROM User WHERE Ort = 'Berlin'; UPDATE User SET Status = 'admin'
WHERE Name = 'Angreifer' --'
```

Damit kann sich der Angreifer Administratorenrechte für das Forum einräumen. Allerdings ist hierfür Wissen über die Datenbankstruktur erforderlich. Informationen über die Datenbankstruktur können eventuell aus Fehlermeldungen gewonnen werden, die von der Datenbank zurückgegeben werden. Ein Beispiel für eine solche Fehlermeldung ist in Abbildung E.1 dargestellt. Aus einer solchen Fehlermeldung heraus kann ein Angreifer nicht nur den genauen Befehl, in den der manipulierte Parameter einfließt, in Erfahrung bringen, sondern auch den entsprechenden SQL-Tabellennamen und diverse Spaltennamen innerhalb dieser Datenbank-Tabelle.

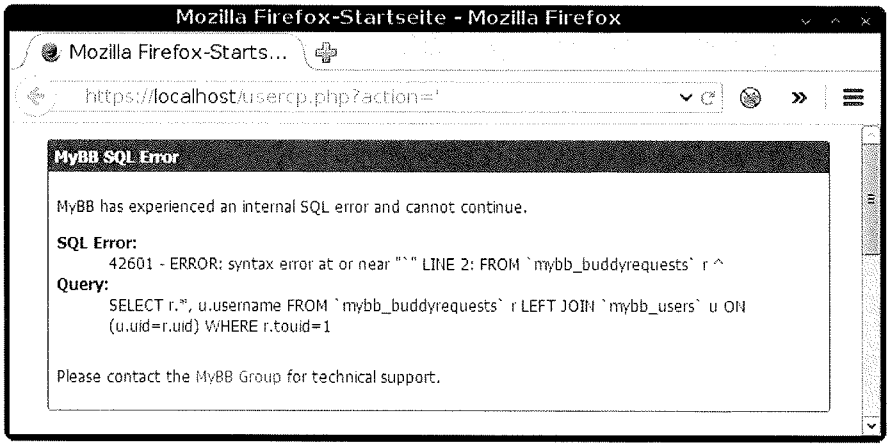

Abbildung E.1: Beispiel einer sehr informativen SQL-Fehlermeldung im Webbrowser.

Ein weiterer Weg, über den ein Angreifer an Informationen über die Datenbankstruktur gelangen kann, ist über die Enumeration von Tabellen- und Spaltennamen mittels in der Datenbank vorhandener Metainformationen[32]. Dieser Vorgang lässt sich mit Tools wie sqlmap[33] auch automatisieren.

Die folgende Eingabe listet die gesamte Tabelle `User` auf, da die Bedingung „1=1" immer erfüllt wird:

```
SELECT * FROM User WHERE Ort = '' OR 1=1 --'
```

Auch andere Programme wie beispielsweise die Eingabeaufforderung unter Microsoft Windows können über einen entsprechenden Aufruf geöffnet werden. Für einen MS-SQL-Server unter Microsoft Windows 2000 ergibt sich hierfür folgende Eingabe:

```
SELECT * FROM User WHERE Ort = '' exec master..xp_cmdshell 'net user angreifer passwort/ADD'
```

Der hintere Teil des Befehls (`xp_cmdshell`) wird vom Server als Extended Stored Procedure erkannt und mitsamt den übergebenen Parametern ausgeführt. In diesem Beispiel wird auf dem System ein Benutzer namens „angreifer" mit dem Passwort „passwort" angelegt.

> ➤ Um das Risiko von SQL Injection-Angriffen zu mindern, wird eine gründliche serverseitige Überprüfung aller Parameter empfohlen, die in irgendeiner Weise in ein SQL-Kommando mit einfließen können. Insbesondere wenn diese Parameter von Benutzern manipulierbar sind, müssen sie einer strengen Filterung unterzogen werden. Besonders empfehlenswert ist hierfür der Einsatz von Prepared Statements für (häufig ausgeführte) SQL-Kommandos, da bei diesen die Parameter automatisch maskiert werden.

32 Als Beispiel sei unter MySQL (in Versionen > 4) die Datenbank `information_schema` genannt, die Informationen über die Datenbankstruktur enthält.

33 `http://sqlmap.sourceforge.net`