Security Advisory

Lara Quitte, Tino Jeromin

1 Programmbeschreibung

Bei dem Programm IT-Sec Forum, welches unter der URL http://127.0.0.1:8081 zu erreichen ist, handelt es sich um eine Webanwendung, die den Austausch unter Fachleuten der IT Sicherheit ermöglichen soll. Bei einer Analyse des Programms konnten Sicherheitslücken festgestellt werden, die im folgenden näher erläutert werden sollen.

2 Vulnerabilität

2.1 Common Vulnerabilty Scoring System

Mithilfe des Calculators auf der Seite https://www.first.org/cvss/calculator/3.1 haben wir den Score unserer der Sicherheitslücke ermittelt. Der Wert liegt hier bei 5.3. Durch das oben genannte Verfahren, ist es möglich die Schwere des Risikos in die folgenden Kategorien einzuteilen: kritisch, hoch, moderat oder niedrig. In unserem Fall entspricht der Wert von 5.3 einem moderaten Risiko. Die Wahl der Attributwerte wurde wie folgt getroffen:

Attributname	Wert	Beschreibung
Attack Vector	Network	Die Website ist aus dem gesamten Internet erreichbar.
Attack Complexity	Low	Der Angriff ist jederzeit und wiederholtmöglich. Es müssen keine besonderen Umstände erfüllt sein.
Privileges Required	None	Es müssen keine besonderen Privilegien vorliegen. Jeder kann diesen Angriff ausführen.
User Interaction	None	Es ist keine Interaktion eines Users nötig.
Scope	Changed	Die angreifbare Komponente, hier die Webseite, ist eine andere, als die Komponente, auf deren Ressourcen man mit einem Angriff zugreift. Die wäre in diesem Fall die Datenbank.
Confidentiality	Low	Der Angreifer kann diverse Daten aus der Datenbank auslesen, einige Daten, wie die MySQL Konfigurationsdatei, sind jedoch weiterhin geschützt.
Integrity	None	Es konnte auf vorliegende Daten zugegriffen werden, diese können allerdings durch einen Angriff nicht modifiziert werden.
Availability	None	Die Webseite ist auch nache einem Angriff noch weiter erreichbar.

2.2 Schadenpotenzial

Wir haben eine Reihe von kritischen Stellen im Programm entdeckt. Diese finden sich genau an den Stellen im Quelltext, wo bereits SQL Anfragen gestellt werden. Ein Angreifer hat hier nun die Möglichkeit durch SQL Injections folgendes zu tun:

- Einloggen als User oder Admin ohne Username und Passwort zu kennen
- Namen der (aktuellen) Datenbank ermitteln
- Namen der Spalten von Tabellen anzeigen
- alle Tabellennamen anzeigen
- gesamte Usertabelle anzeigen
- MySQL Systemvariablen ausgeben lassen

Genauere Informationen zu den angeführten SQL Injections, sowie der Erklärung diverser SQL Anfragen finden sich unter den Links https://www.w3schools.com/sql/, https://dev.mysql.com/doc/refman sowie auch auf vielen anderen Webseiten, die im Internet zu finden sind.

3 Verringerung des Schadens

Um mehr Sicherheit zu gewährleisten und das Risiko von SQL Injections zu verringern, empfehlen wir die folgenden Optionen:

3.1 Option 1: Input Validation

Die Eingabe von Usern kann gefiltert werden, um unerwünschte Eingaben zu verhindern. Dies kann zum Beispiel über reguläre Ausdrücke für Name, Alter, etc. gelöst werden. Weiterhin wäre es eine Möglichkeit die Eingabe über eine Drop-Down-Liste zu gestalten, so dass keine direkte Nutzereingabe gefordert ist.

3.2 Option 2: Parameterized Queries

Dies sind vorgefertigte SQL Anfragen, bei denen durch die Parameter die Nutzereingaben eingefügt werden können. So kann die Datenbank den Quellcode von der Nutzereingabe unterscheiden und eine missbräuchliche Verwendung der Nutzereingabe wird verhindert. MySQL-Connector bietet eine solche Möglichkeit in automatisierter Form.

3.3 Option 3: Character Escaping Functions

Für Nutzereingaben sollten immer Funktionen mit Character Escaping genutzt werden, welche vom Datenbank Management System bereitgestellt werden. Das bedeutet, dass bei einer Nutzereingabe bestimmte Zeichen, wie z.B einfache/doppelte Anführungszeichen mit einem vorangestellten Backslash markiert werden, so dass diese von der eigentlichen Nutzereingabe zu unterscheiden sind.

Die erforderlichen Änderungen des Quellcodes mit enthaltener Fehlerbehebung findet sich in den untenstehenden Abbildungen. Dafür wurden die abgebildeten Funktionen modifiziert. Hier wurde die SQL Anfrage in der Art verändert, dass eine SQL Injection nicht mehr möglich ist. Bei dieser Änderung handelt es sich um eine Verwendung der oben genannten Option 2, der Parameterized Queries.

```
def checkCredentials(username, password):
    '''Check the provided credentials and return the username or false'''
    result = False
    con, cursor = None, None
    try:
        con = connect()
        # Statt den username und password in den query-String zu kopieren, wird ein Format-String
        # verwendet, um die Nutzereingaben als Parameter zu bergeben .
        # Dies ist die von MySQL empfohlene Variante.
        query = "SELECT username FROM users WHERE username = %s AND password = %s"
        cursor = con.cursor()
        cursor.execute(query, username, password)
        rows = cursor.fetchall()
        if rows:
            result = rows[0][0]
    except mysql.connector.Error as e:
        print(e)
    finally:
        if cursor: cursor.close()
        if con: con.close()
    return result
```

Figure 1: itsecforum/database.py

```
def isAdmin(username):
    '''Check if provided username is admin'''
    result = False
    con, cursor = None, None
    try:
        con = connect()
        query = "SELECT username FROM users WHERE username = %s AND admin = 1"
        cursor = con.cursor()
        cursor.execute(query, username)
        rows = cursor.fetchall()
        if rows:
            result = True
    except mysql.connector.Error as e:
        print(e)
    finally:
        if cursor: cursor.close()
        if con: con.close()
    return result
```

Figure 2: itsecforum/database.py

```
def getEmail(username):
    '''Return the email address for the given username or false'''
    result = False
    con, cursor = None, None
    try:
        con = connect()
        query = "SELECT mail FROM users WHERE username = %s"
        cursor = con.cursor()
        cursor.execute(query, username)
        rows = cursor.fetchall()
        if rows:
            result = rows[0][0]
    except mysql.connector.Error as e:
        print(e)
    finally:
        if cursor: cursor.close()
        if con: con.close()
    return result
```

Figure 3: itsecforum/database.py

4 Fazit

Die Grundlage von SQL Injections besteht hauptsächlich aus der Möglichkeit für den Angreifer, Abfragen auf Datenbanken so zu verändern, dass die eigentliche Funktion der Abfrage verändert wird. Der Angreifer hat so die Möglichkeit Daten abzufragen, zu manipulieren oder zu löschen und im schlimmsten Fall auch die vollständige Kontrolle über die Datenbank zu übernehmen. Mithilfe der Optionen, welche in Kapitel 3 vorgestellt wurden, kann ein solcher Angriff verhindert werden oder es wird zumindest das Risko für solche Attacken verringert. Die gefundenen Sicherheitslücken sind mit wenigen Schritten zu beheben und die Webseite kann so schnell sicherer gemacht werden, wie in den Abbildungen (Figure 1 - Figure 3) ersichtlich wird.