NoSQL-Injection Angriffe verhindern

In diesem Abschnitt werden verschiedene Methoden erläutert, wie NoSQL-Injection Angriffe verhindert werden können. Dabei werden verschiedene Methoden erläutert und spezifische Beispiele für eine MongoDB Datenbank in PHP-Umgebung gegeben.

## Methoden:

1. Search and Replace: Durch das Überprüfen der Benutzereingabe kann verhindert werden, dass Datenbankoperatoren (wie z.B. $where) ausgeführt werden. Dafür kann die Benutzereingabe auf nicht gewünschte Zeichen wie „$“ oder andere Sonderzeichen geprüft werden, um diese dann aus der Eingabe zu entfernen. In einer PHP-Umgebung wäre dies mit der Funktion preg\_replace möglich [1].
2. Type Casting: Beim Type Casting wird die Eingabe des Benutzers in den gewünschten Datentyp umgewandelt. Bei der Eingabe von LogIn-Daten wäre dies bspw. eine Umwandlung der Benutzereingabe in den Datentyp String. In einer PHP-Umgebung wäre dies mit der Funktion type-juggling möglich [2][6].
3. Prepared Statements: Benutzereingaben sollten getrennt von der Datenbankabfrage verwendet werden, um zu verhindern das die Benutzereingabe von der Datenbank als Teil der Abfrage interpretiert wird. Mögliche Datenbankoperatoren aus der Benutzereingabe werden so nicht in der Datenbankabfrage ausgeführt [3].
4. Least-Privilige-Prinzip: Benutzeraccounts sollten die geringstmöglichen Zugriffsrechte zur Datenbank haben. Durch eine sorgfältige Verteilung von Zugriffsrechten, kann bei einem erfolgreichen Angriff der mögliche Schaden reduziert werden, da Angreifer durch die niedrige Zugriffsebene auf weniger Daten Zugriff haben [3][4].
5. Whitelist: Ähnlich zur Search and Replace Methode kann auch eine Whitelist mit zulässigen Zeichen erstellt werden, um Angriffe zu verhindern. Bei einer Whitelist werden die Benutzereingaben auf zulässige Zeichen überprüft. Eingaben mit unzulässigen Zeichen werden erkannt und im Anschluss nicht ausgeführt.
6. Aktuelle Versionen: Beim Implementieren einer Anwendung sollte darauf geachtet werden die neuesten stabilen Versionen der verwendeten Architektur zu verwenden, um interne Sicherheitslücken in der Architektur zu vermeiden.
7. Pentesting/Security Audits: Eine weitere Möglichkeit ist das Nutzen von Security Audits. Diese Tools überprüfen die Einstellungen der Datenbank auf bekannte Fehler und geben Empfehlungen wie man seine Datenbank sicherer gestalten kann. Außerdem können einige dieser Tools auch Pentesting durchführen, um Schwachstellen in der Architektur zu finden. Für MongoDB wäre das bspw. MongoAudit [10].
8. Libraries/Packages: Es gibt auch bereits Libaries oder Packages, die mehrere der vorher genannten Methoden umsetzen. Für MongoDB wären das bspw. mongo-sanitize oder mongoose, welche die Benutzereingaben überprüfen und Type Casting oder Search and Replace beinhalten [7][8][9].

## Leitfaden (Umsetzung anhand der Implementierung als Beispiel)

1. Zuerst wird die unsichere Version gezeigt.
2. Nutzer können dann im zweiten Schritt einzelne Methoden sehen. (Beispiel: Type Casting, S earch and Replace)
3. Paketbeispiele für MongoDB: z.B. Mongoose [8], mongo-sanitize [7]

## Beispiele für MongoDB auf PHP-Server

1. Pentesting & Security Audits (Mongoaudit) [5][10]
2. Pakete für die Umsetzung der Methoden (Mongoose, etc.)

Quellen

[1] Search and Replace in PHP: <https://www.php.net/manual/en/function.preg-replace.php>

[2] Type Casting in PHP: <https://www.php.net/manual/en/language.types.type-juggling.php>

[3] <https://infosecwriteups.com/nosql-injection-8732c2140576>

[4] <https://www.cyberark.com/de/what-is/least-privilege/>

[5] <https://geekflare.com/de/nosql-security-scan/>

[6] <https://owasp.org/www-pdf-archive/GOD16-NOSQL.pdf>

[7] <https://www.npmjs.com/package/mongo-sanitize>

[8] <https://mongoosejs.com/>

[9] <https://appsecco.com/blog/hacking-apps-using-nosql-injection>

[10] <https://github.com/stampery/mongoaudit>