Engineer

SQL-Injektion

Angriff, der Eingaben zu Teilen einer Datenbankabfrage macht

Was es ist

SQL-Injektion, oft mit SQLi abgekürzt, ist ein Angriff, bei dem eine Anwendung einen SQL-Befehl aus fremden Texteingaben zusammensetzt und die Datenbank einen Teil dieses Textes als SQL statt als Wert liest. Die Common Weakness Enumeration führt sie als CWE-89 und beschreibt die Schwachstelle als einen Befehl, der ganz oder teilweise aus externen Eingaben aufgebaut wird, ohne zuvor die Zeichen zu neutralisieren, die in SQL eine besondere Bedeutung haben. Das Präventionsmerkblatt von OWASP beschreibt denselben Zusammenhang einfacher: Der Angriff funktioniert, wenn nicht vertrauenswürdige Daten Teil der Abfragesyntax werden.

Die Ursache liegt darin, dass SQL keine feste Grenze zwischen den Anweisungen eines Befehls und den darin enthaltenen Daten zieht. Ein einfaches Anführungszeichen beendet ein Zeichenkettenliteral; was der Angreifer danach eingibt, wird als weiteres SQL geparst. Zwei Bindestriche beginnen einen Kommentar, mit dem der Angreifer den Rest der vom Entwickler geschriebenen Anweisung verwerfen kann. Ein Semikolon beendet eine Anweisung und beginnt eine weitere, sofern Datenbank und Treiber dies zulassen. Keines dieser Zeichen ist ungewöhnlich: Sie kommen auch in Nachnamen und Freitext vor.

Der Web Security Testing Guide von OWASP teilt die Angriffe danach in drei Klassen ein, wie die Ergebnisse zum Angreifer gelangen. Bei In-Band-Angriffen wird die Antwort über denselben Kanal gelesen, über den die Eingabe gesendet wurde, meist auf der Seite der Anwendung selbst. Bei Out-of-Band-Angriffen liefert die Datenbank die Ergebnisse an anderer Stelle ab, etwa über eine separate HTTP-Verbindung. Inferenzbasierte oder blinde Angriffe erhalten überhaupt keine Daten und rekonstruieren sie Frage für Frage aus dem Verhalten der Anwendung. Der Leitfaden nennt fünf verbreitete Techniken innerhalb

dieser Klassen: einen `UNION` -Operator, der ein zweites `SELECT` mit dem ersten verbindet, boolesche Bedingungen, absichtlich ausgelöste Fehler, Out-of-Band-Anfragen und gezielte Zeitverzögerungen.

SQL-Injektion gehört zu einer größeren Familie. In CWE ist sie der allgemeinen Injektionsschwachstelle CWE-74 zugeordnet, neben Cross-Site-Scripting; die OWASP Top Ten für 2025 ordnen beide der Kategorie A05 Injection zu. Gemeinsam ist ihnen, dass Daten eine Grenze überschreiten und an einer Stelle als Anweisungen gelesen werden. Hier ist der Interpreter der SQL-Parser der Datenbank. Die Abhilfe besteht darin, sämtliche Werte vollständig aus dem SQL-Text herauszuhalten.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist eine SQL-Injektion?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Der Schaden hängt davon ab, was das Datenbankkonto der Anwendung tun darf, und das ist häufig sehr viel. CWE-89 nennt das Lesen von Daten, die der Angreifer niemals sehen sollte, das Umgehen der Authentifizierung ohne Kenntnis eines Passworts, das Ändern der Daten, die festlegen, wer was darf, das Ändern oder Löschen von Datensätzen und in manchen Konfigurationen das Ausführen von Betriebssystembefehlen. Die OWASP-Seite über den Angriff ergänzt administrative Vorgänge wie das Herunterfahren der Datenbank, das Lesen von Dateien auf dem Datenbankserver und die Übernahme des Datenbankservers mit Administratorrechten.

Beide Quellen stufen die Schwachstelle als schwerwiegend ein. CWE bewertet die Wahrscheinlichkeit einer Ausnutzung als hoch; die OWASP-Seite über den Angriff beschreibt den Fehler als leicht auffindbar und leicht ausnutzbar. Die Schwachstelle erscheint in jeder Ausgabe der CWE Top 25 Most Dangerous Software Weaknesses von 2019 bis 2025. Der CWE-Eintrag dokumentiert außerdem eine Angriffswelle von 2008, bei der eine einzige SQL-

Injektionszeichenfolge gegen viele verschiedene Programme eingesetzt wurde, zahlreiche Webserver kompromittierte und sie zur Auslieferung schädlichen Codes brachte.

Die Abwehr ist dagegen gut verstanden und kostengünstig. Das OWASP-Merkblatt nennt vorbereitete Anweisungen mit gebundenen Parametern als erste Schutzmaßnahme; CWE empfiehlt sie ebenfalls: SQL und Werte werden getrennt übertragen, sodass die Datenbank das eine nicht mit dem anderen verwechseln kann. Alles Weitere, vom Prinzip der geringsten Rechte bis zur sorgfältigen Fehlerbehandlung, bildet eine zweite Schutzschicht, die den Schaden begrenzt, wenn doch ein Fehler durchrutscht.

Wie es in der Praxis aussieht

Nehmen wir einen Shop, dessen Bestellhistorie Bestellungen anhand einer in ein Formular eingegebenen E-Mail-Adresse abfragt. Der Entwickler baut das SQL mit einem Python-f-String:

```
cur.execute(f"SELECT id, total FROM orders WHERE email = '{email}'")
```

Ein Besucher gibt `x' OR '1'='1` in das Feld ein. Das Programm fügt den Text ein, und die Datenbank erhält Folgendes:

```
SELECT id, total FROM orders WHERE email = 'x' OR '1'='1'
```

Das Anführungszeichen nach `x` schließt die vom Entwickler geöffnete Zeichenkette. Die darauf folgende `OR` -Bedingung ist immer wahr, sodass die Datenbank jede Bestellung in der Tabelle zurückgibt. CWE-89 und der Testleitfaden verwenden denselben Trick mit einer immer wahren Bedingung. Damit kann auch ein Anmeldeformular einen Besucher akzeptieren, der weder Benutzername noch Passwort kennt.

Eine noch schädlichere Eingabe beendet die Anweisung und beginnt eine weitere. Im Beispiel des CWE-Eintrags wird `name'; DELETE FROM items; --` als Artikelname eingegeben. Dadurch folgt auf die erste Anweisung eine zweite, und die Bindestriche kommentieren den Rest aus. Ob dies ausgeführt wird, hängt von Datenbank und Treiber ab; mehrere aktuelle Schnittstellen lehnen es ab.

PostgreSQLs `PQexecParams` akzeptiert höchstens einen SQL-Befehl pro Zeichenkette, was die Dokumentation als einen gewissen Schutz gegen SQL-Injektion einordnet. Pythons Modul `sqlite3` löst einen Fehler aus, sobald `execute()` mehr als eine Anweisung erhält. Psycopg lehnt mehrere Anweisungen in einem Aufruf ab, wenn Parameter übergeben werden. Nichts davon verhindert die oben gezeigte immer wahre Bedingung: Sie benötigt nur eine Anweisung.

Die Lösung besteht darin, den Wert separat zu übergeben. PEP 249, der Standard für Pythons Datenbankschnittstelle, verlangt von jedem Treiber, seinen Platzhalterstil in einer Modulkonstante namens `paramstyle` anzugeben, und nennt fünf Stile: `qmark` (`?`), `numeric` (`:1`), `named` (`:name`), `format` (`%s`) und `pyformat` (`%(name)s`). Das Modul `sqlite3` ist auf `qmark` festgelegt und akzeptiert auch `named` . Psycopg, der PostgreSQL-Treiber, verwendet `%s` und `%(name)s` :

```
cur.execute(
    "SELECT id, total FROM orders WHERE email = %s",
    (email,),
)
```

Das `%s` sieht wie Pythons Zeichenkettenformatierung aus, doch hier wird nichts formatiert: Das SQL und das Tupel mit den Werten erreichen den Treiber als zwei getrennte Argumente. Psycopg 3 hält sie auf dem gesamten Weg auseinander und sendet den Befehl und seine Parameter getrennt an den Server, statt sie auf dem Client zusammenzuführen. Das OWASP-Merkblatt verdeutlicht dies mit seiner eigenen Beispieleingabe `tom' or '1'='1` : Als Parameter gebunden ist sie ein ungewöhnlicher Benutzername, nach dem wörtlich gesucht wird und der keinem Benutzer entspricht.

PostgreSQL bietet dieselbe Trennung auch in reinem SQL. `PREPARE` legt die Anweisung mit nummerierten Parametern fest; `EXECUTE` liefert die Werte später:

```
PREPARE order_lookup (text) AS
    SELECT id, total FROM orders WHERE email = $1;
EXECUTE order_lookup('x'' OR ''1''=''1');
```

Die feindliche Eingabe, hier als SQL-Literal geschrieben, kommt in `$1` als eine einzige merkwürdige E-Mail-Adresse an und findet keine Bestellungen.

Auf einer Plattform mit beruflichen und organisatorischen Daten werden Eingaben validiert, bevor irgendetwas auf die Datenbank zugreift. Anschließend verbindet sich die API mit einer Rolle nach dem Prinzip der geringsten Rechte, die nur die Funktionen der Anwendung ausführen darf; sämtliche Werte werden parametrisiert. Die Bindung verhindert, dass Daten zu SQL werden, und die eng begrenzte Rolle beschränkt, was ein Fehler erreichen könnte.

Wo es schiefgeht

Die Zeichenkette trotzdem zusammensetzen. Ein f-String, der Operator `%`, `+` oder `.format()` fügt den Wert in das SQL ein, bevor der Treiber es sieht. Dasselbe gilt für einen in Anführungszeichen geschriebenen Platzhalter wie `'%s'`. OWASP weist darauf hin, dass das Verketten nicht vertrauenswürdiger Werte vor dem Vorbereiten einer Anweisung den gesamten Schutz aufhebt. Die Psycopg-Dokumentation fordert Entwickler auf, niemals `%` oder `+` zum Einfügen von Werten in Abfragen zu verwenden. Übergeben Sie jeden Wert als zweites Argument von `execute()`, ohne Anführungszeichen um den Platzhalter.

Von Hand maskieren. Anführungszeichen zu verdoppeln oder Backslashes einzufügen wirkt gleichwertig, doch die Regeln hängen vom SQL-Dialekt und seinen Einstellungen ab. OWASP nennt als Beispiel MySQLs Modus `NO_BACKSLASH_ESCAPES`, der die Maskierung durch Backslashes abschaltet; das Merkblatt rät nachdrücklich vom Maskieren als Schutzmaßnahme ab. Ersetzen Sie von Hand maskierte Zeichenketten durch gebundene Parameter.

Validierung oder eine WAF als Lösung betrachten. Validierung fängt Fehler ab, doch Freitext muss die Zeichen zulassen, die ein Angreifer braucht: Der CWE-Eintrag nennt den Nachnamen O’Reilly, der eine Validierung wahrscheinlich bestehen würde und trotzdem ein Anführungszeichen enthält. Sperrlisten gefährlicher Wörter haben Lücken, und CWE stuft die Wirksamkeit einer Anwendungsfirewall nur als mäßig ein. OWASP formuliert es so: „Die Validierung eines Werts macht es nicht sicher, ihn mit SQL zu verketten.“ Validieren Sie weiterhin auf Korrektheit und binden Sie Werte für die Sicherheit.

Tabellen- oder Spaltennamen oder die Sortierreihenfolge aus Eingaben übernehmen. Platzhalter stehen nur für Werte. Ein Entwickler, der Benutzer eine Spalte oder die Richtung von `ORDER BY` wählen lassen muss, ist deshalb versucht,

den Text einzufügen. OWASPs Antwort lautet, jede Auswahl auf eine kleine Menge anwendungseigener Namen abzubilden und alles andere abzulehnen, beispielsweise eine gewünschte Richtung auf das Literal `ASC` oder `DESC`. Wenn ein Name tatsächlich dynamisch sein muss, setzt Psycopgs Modul `psycopg.sql` Bezeichner zusammen. OWASP weist jedoch darauf hin, dass das korrekte Einschließen eines Namens in Anführungszeichen nicht begrenzt, auf welche Tabellen sich die Anwendung lenken lässt.

Gespeicherte Prozeduren für sicher halten. Eine Prozedur mit gebundenen Argumenten aufzurufen schützt den Aufruf, nicht das SQL, das die Prozedur intern zusammenbaut. CWE weist darauf hin, dass eine gespeicherte Prozedur nur einige Injektionswege ausschließt, während andere möglich bleiben. Verwenden Sie innerhalb der Datenbank statisches SQL oder binden Sie Werte auch in dynamischem SQL; in PostgreSQL bedeutet das `EXECUTE ... USING` für Werte und `format('%I', ...)` für Bezeichner.

Ein ORM mit eingefügten Zeichenketten. Ein objektrelationaler Mapper bindet Werte, wenn er wie vorgesehen verwendet wird. CWE bescheinigt Persistenzschichten wie Hibernate bei korrekter Verwendung einen erheblichen Schutz. OWASP ergänzt, dass auch Hibernate-Abfragen für Injektionen anfällig sind, wenn Entwickler Eingaben mit dem Abfragetext verketten. Verwenden Sie die Parameter des Mappers, statt Zeichenketten zusammenzubauen.

Ein Anwendungskonto mit Administratorrechten. Kann das Konto alles, kann es auch der Angreifer. OWASP fordert schreibgeschützte Konten, wenn nur Lesezugriffe benötigt werden, einen eigenen Datenbankbenutzer für jede Webanwendung, Sichten, die nur die vom Konto benötigten Spalten freigeben, und keine Administratorrechte für Anwendungskonten. Arbeitet eine Anwendung ausschließlich über gespeicherte Prozeduren, sollte sie nur diese ausführen dürfen und sonst nichts. Das Prinzip der geringsten Rechte begrenzt den Schaden; es verhindert die Injektion nicht.

Ausführliche Datenbankfehler Benutzern anzeigen. Eine Fehlermeldung, die die Struktur der Anweisung offenlegt, hilft dem Angreifer, den nächsten Versuch anzupassen. Genau darauf beruht die fehlerbasierte Technik des Testleitfadens. CWE empfiehlt knappe Fehlermeldungen. Zeigen Sie Benutzern einen allgemeinen Fehler und behalten Sie die Details für das Team auf dem Server.

Bereits gespeicherten Daten vertrauen. Ein sicher gespeicherter Wert kann weiterhin gefährlich sein, wenn späterer Code ihn in SQL einfügt. Laut OWASP gelten die Schutzmaßnahmen unabhängig davon, woher Daten stammen: aus einer HTTP-Anfrage, einem anderen Dienst oder einem früheren Schreibvorgang. CWE warnt, dass spätere Prozesse die Maskierung gespeicherter Werte vergessen können. Binden Sie Werte aus Ihren eigenen Tabellen genauso wie Eingaben aus einem Formular.

Weiterführende Lektüre

- CWE-89 in der Common Weakness Enumeration — der Schwachstelleneintrag: Folgen, Beispiele und Schutzmaßnahmen mit ihren Grenzen.
- OWASP SQL Injection Prevention Cheat Sheet — parametrisierte Abfragen, gespeicherte Prozeduren, dynamische Bezeichner und das Prinzip der geringsten Rechte.
- OWASP Web Security Testing Guide: Testing for SQL Injection — Angriffsklassen und Techniken aus der Sicht eines Testers.
- PEP 249: Python Database API Specification v2.0 — die gemeinsame Schnittstelle der Python-Treiber einschließlich der fünf Platzhalterstile.
- Psycopg 3: Passing parameters to SQL queries — die Regeln für Platzhalter und das richtige Zusammensetzen dynamischen SQLs.
- PostgreSQL: Command Execution Functions — wie libpq Parameter getrennt von der Befehlszeichenkette überträgt.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist eine SQL-Injektion?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/sql-injection/