Engineer

SQL-injection

et angreb, der gør input til en del af en SQL-forespørgsel

Hvad det er

SQL-injection, på engelsk skrevet SQL injection og ofte forkortet SQLi, er det angreb, hvor en applikation bygger en SQL-kommando af tekst, som en anden har leveret, og databasen ender med at læse en del af teksten som SQL i stedet for som en værdi. Common Weakness Enumeration fører det som CWE-89 og beskriver svagheden som en kommando, der helt eller delvis er bygget af input udefra, uden at de tegn, der har særlig betydning i SQL, først er blevet neutraliseret. OWASP's cheat sheet om forebyggelse siger det mere ligefremt: angrebet virker, når data, man ikke kan stole på, bliver en del af syntaksen i en forespørgsel.

Roden til problemet er, at SQL ikke trækker nogen skarp grænse mellem instruktionerne i en kommando og dataene i den. Et enkelt anførselstegn afslutter en strengkonstant, og det, angriberen skriver efter det, bliver tolket som mere SQL. To bindestreger indleder en kommentar, så angriberen kan skaffe sig af med resten af udviklerens sætning. Et semikolon afslutter, hvor databasen og driveren tillader det, én sætning og begynder en ny. Ingen af tegnene er eksotiske: de samme tegn dukker op i efternavne og fritekst.

OWASP's Web Security Testing Guide deler angrebene i tre klasser efter, hvordan resultaterne når frem til angriberen. In-band-angreb læser svaret gennem den samme kanal, som de brugte til at sende det, typisk på applikationens egen side. Out-of-band-angreb får databasen til at levere resultaterne et andet sted hen, for eksempel over en separat HTTP-forbindelse. Inferentielle eller blinde angreb modtager slet ingen data og stykker dem sammen et spørgsmål ad gangen ud fra, hvordan applikationen opfører sig. Guiden nævner fem almindelige teknikker på tværs af de klasser: en `UNION` -operator, der hægter en ekstra `SELECT` på den første, booleske betingelser, fejl fremprovokeret med vilje, en separat out-of-band-kanal og bevidste tidsforsinkelser.

SQL-injection hører til en større familie. I CWE står den under den generelle injection-svaghed, CWE-74, side om side med cross-site scripting, og OWASP Top Ten for 2025 placerer begge i kategorien A05 Injection. Den røde tråd er data, der krydser en grænse og havner et sted, hvor de bliver læst som instruktioner. Her er fortolkeren databasens SQL-parser, og kuren er at holde hver eneste værdi helt ude af SQL-teksten.

Brug begrebet i en beslutning

Brug »Hvad er SQL-injection?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Konti, adgangskoder og rettigheder sat op, så folk har præcis den adgang, jobbet kræver, og fratrådte mister den samme dag.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

Skaden afhænger af, hvad applikationens databasekonto må, og det er ofte meget. CWE-89 nævner, at angriberen kan læse data, der aldrig skulle have været synlige, komme forbi autentificeringen uden at kende en adgangskode, ændre de data, der afgør, hvem der må hvad, ændre eller slette poster og i nogle opsætninger køre kommandoer i styresystemet. OWASP's side om angrebet tilføjer administrative handlinger som at lukke databasen ned, læse filer fra databaseserveren og overtage databaseserveren som dens administrator.

Begge kilder regner det for alvorligt. CWE angiver sandsynligheden for udnyttelse som høj, og OWASP's side om angrebet beskriver fejlen som både let at finde og let at udnytte. Svagheden står i hver udgave af CWE Top 25 Most Dangerous Software Weaknesses fra 2019 til 2025. CWE-posten beretter også om en bølge i 2008, hvor én og samme angrebsstreng, der virkede mod mange forskellige programmer, kompromitterede et stort antal webservere og fik dem til at servere ondsindet kode.

Forsvaret er til gengæld velkendt og billigt. OWASP's cheat sheet gør prepared statements med bundne parametre til sit første forsvar, og CWE anbefaler dem også: SQL'en og værdierne sendes hver for sig, så databasen ikke kan forveksle det ene med det andet. Alt andet, fra mindste rettigheder til omhyggelig fejlhåndtering, er et ekstra lag, der begrænser skaden, når en fejl slipper igennem.

Sådan ser det ud i praksis

Tag en webshop, hvis side med ordrehistorik slår ordrer op ud fra den e-mailadresse, kunden taster ind i en formular. Udvikleren bygger SQL'en med en f-streng i Python:

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

En besøgende skriver `x' OR '1'='1` i feltet. Programmet sætter det direkte ind, og databasen modtager dette:

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

Anførselstegnet efter `x` lukker den streng, udvikleren åbnede, og `OR`-leddet bagefter er altid sandt, så databasen returnerer hver eneste ordre i tabellen. CWE-89 og testguiden bruger det samme altid sande trick, som også kan få en loginformular til at lukke en besøgende ind, der hverken kender brugernavnet eller adgangskoden.

Et værre input afslutter sætningen og begynder en ny. Eksemplet i CWE-posten indtaster `name'; DELETE FROM items; --` som varenavn, så en ekstra sætning følger efter den første, og bindestregerne gør resten til en kommentar. Om det kører, afhænger af databasen og driveren, og flere nutidige grænseflader afviser det. PostgreSQL's `PQexecParams` tager højst én SQL-kommando pr. streng, hvilket dokumentationen regner for et vist ekstra værn mod SQL-injection. Pythons `sqlite3`-modul rejser en fejl, når `execute()` får mere end én sætning, og psycopg afviser flere sætninger i ét kald, når der sendes parametre med. Intet af det stopper den altid sande betingelse ovenfor, som kun kræver én sætning.

Løsningen er at sende værdien for sig. PEP 249, Pythons standard for databasegrænseflader, lader hver driver angive sin pladsholderstil i en modulkonstant ved navn `paramstyle` og nævner fem stilarter: `qmark` (`?`), `numeric` (`:1`), `named` (`:name`), `format` (`%s`) og `pyformat` (`%(name)s`). `sqlite3`-modulet er låst til `qmark` og accepterer også `named`. Psycopg, driveren til PostgreSQL, tager `%s` og `%(name)s`:

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

`%s` ligner strengformatering i Python, men intet bliver formateret: SQL'en og tuplen med værdier når frem til driveren som to separate argumenter. Psycopg 3 holder dem adskilt hele vejen og sender kommandoen og dens parametre hver for sig til serveren i stedet for at flette dem sammen på klienten. OWASP's cheat sheet viser pointen med sit eget eksempel på input, `tom' or '1'='1` : bundet som parameter er det bare et mærkeligt brugernavn, der slås op bogstaveligt, og det passer på ingen.

PostgreSQL tilbyder den samme opdeling i ren SQL. `PREPARE` fastlægger sætningen med nummererede parametre, og `EXECUTE` leverer værdierne bagefter:

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

Det fjendtlige input, her skrevet som en SQL-strengkonstant, ankommer i `$1` som én underlig e-mailadresse og finder ingen ordrer.

På en platform med professionelle og organisatoriske data bliver input valideret, før noget når databasen, og derefter forbinder API'et sig som en least-privilege-rolle, der kun må køre app-funktionerne, og alt er parametriseret. Bindingen forhindrer data i at blive til SQL, og den snævre rolle begrænser, hvad en fejl kan nå.

Hvor det går galt

At bygge strengen alligevel. En f-streng, `%` -operatoren, `+` eller `.format()` fletter værdien ind i SQL'en, før driveren ser den, og det samme gør en pladsholder skrevet inde i anførselstegn, som `'%s'` . OWASP påpeger, at hvis værdier, man ikke kan stole på, kædes sammen med SQL'en, før sætningen forberedes, er hele beskyttelsen sat ud af kraft, og psycopg's dokumentation

siger til udviklere, at de aldrig må bruge `%` eller `+` til at flette værdier ind i forespørgsler. Send hver værdi som andet argument til `execute()`, med pladsholderen uden anførselstegn.

Escaping i hånden. At fordoble anførselstegn eller tilføje backslashes ser ud til at gøre det samme, men reglerne afhænger af SQL-dialekten og dens indstillinger. OWASP's eksempel er MySQL's tilstand `NO_BACKSLASH_ESCAPES`, der slår escaping med backslash fra, og cheat sheetet fraråder escaping på det kraftigste. Erstat håndescapede strenge med bundne parametre.

Validering eller en WAF som løsningen. Validering fanger fejl, men fritekst skal tillade de tegn, en angriber har brug for: CWE-postens eksempel er efternavnet O'Reilly, som formentlig ville gå igennem valideringen og stadig indeholder et anførselstegn. Lister over forbudte ord har huller, og CWE vurderer effekten af en applikationsfirewall som kun moderat. OWASP siger det ligeud: at en værdi er valideret, gør den ikke sikker at sætte sammen med SQL. Bliv ved med at validere for korrekthedens skyld, og bind for sikkerhedens.

Tabelnavne, kolonnenavne eller sorteringsrækkefølge hentet fra input. Pladsholdere står kun for værdier, så en udvikler, der skal lade brugeren vælge en kolonne eller retningen i `ORDER BY`, fristes til at sætte teksten direkte ind. OWASP's svar er at oversætte hvert valg til et lille sæt navne, som applikationen selv ejer, og afvise alt andet, for eksempel ved at gøre en ønsket retning til den bogstavelige `ASC` eller `DESC`. Når et navn virkelig skal være dynamisk, kan psycopg's modul `psycopg.sql` sammensætte identifikatorer, men OWASP bemærker, at selv et korrekt citeret navn ikke begrænser, hvilke tabeller applikationen kan lokkes til at bruge.

Stored procedures, der antages at være sikre. At kalde en procedure med bundne argumenter beskytter kaldet, ikke den SQL, proceduren selv bygger indeni. CWE bemærker, at stored procedures forhindrer nogle typer SQL-injection, men ikke mange andre. Brug statisk SQL inde i databasen, eller bind også værdierne i dynamisk SQL; i PostgreSQL betyder det `EXECUTE ... USING` til værdier og `format('%I', ...)` til identifikatorer.

En ORM med strenge syet ind. En objektrelationel mapper binder værdier, når den bruges som tænkt, og CWE tilskriver persistenslag som Hibernate en betydelig beskyttelse, når de bruges korrekt. OWASP tilføjer, at Hibernate-forespørgsler stadig kan angribes, når udviklere sætter input sammen med forespørgselsteksten. Brug mapperens parametre, ikke strengbygning.

En applikationskonto med administratorrettigheder. Hvis kontoen kan alt, kan angriberen også. OWASP beder om skrivebeskyttede konti, hvor der kun skal læses, en separat databasebruger til hver webapplikation, views, der kun viser de kolonner, en konto har brug for, og ingen administratorrettigheder til applikationskonti. Når en applikation kun arbejder gennem stored procedures, bør den have lov til at køre dem og intet andet. Mindste rettigheder begrænser skaden; de forhindrer ikke selve angrebet.

Detaljerede databasefejl vist til brugerne. En fejlmeddelelse, der afslører sætningens opbygning, hjælper en angriber med at tilpasse næste forsøg, og testguidens fejlbaserede teknik bygger netop på det. CWE anbefaler minimale fejlmeddelelser. Vis brugeren en generisk fejl, og hold detaljerne på serveren til teamet.

Blind tillid til data, der allerede ligger i databasen. En værdi, der blev gemt sikkert, kan stadig være farlig, når et senere stykke kode sætter den ind i SQL. OWASP siger, at dens kontroller gælder, uanset om data kommer fra en HTTP-request, en anden tjeneste eller tidligere gemte data, og CWE advarer om, at senere processer kan glemme at escape det, der blev gemt. Bind værdier, der læses fra jeres egne tabeller, ligesom I binder input fra en formular.

Læs mere

- CWE-89 in the Common Weakness Enumeration — svaghedsposten: konsekvenser, eksempler og afhjælpningerne med deres begrænsninger.
- OWASP SQL Injection Prevention Cheat Sheet — parametriserede forespørgsler, stored procedures, dynamiske identifikatorer og mindste rettigheder.
- OWASP Web Security Testing Guide: Testing for SQL Injection — angrebsklasserne og teknikkerne, som en tester møder dem.
- PEP 249: Python Database API Specification v2.0 — den grænseflade, Pythons drivere deler, herunder de fem pladsholderstile.
- Psycopg 3: Passing parameters to SQL queries — reglerne for pladsholdere og den rigtige måde at bygge dynamisk SQL på.
- PostgreSQL: Command Execution Functions — hvordan libpq sender parametre adskilt fra selve kommandostrengen.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er SQL-injection?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/sql-injection/