Внедрение SQL-кода

Внедрение SQL-кода (англ. SQL injection) — один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.

Принцип атаки внедрения SQL

Допустим, серверное ПО, получив входной параметр id, использует его для создания SQL-запроса. Рассмотрим следующий PHP-скрипт:

```
# Предыдущий код скрипта...
$id = $_REQUEST['id'];
$res = mysql_query("SELECT * FROM news WHERE id_news =".$id);
# Следующий код скрипта...
```

Если на сервер передан параметр id, равный 5

(например так: http://example.org/script.php?id=5),

то выполнится следующий SQL-запрос:s WHERE

id_news = 5

SELECT * **FROM** news **WHERE** id_news = 5

Если злоумышленник передаст в качестве параметра id строку -1 OR 1=1 (например, так: http://example.org/script.php?id=-1+OR+1=1), то выполнится запрос:

SELECT * **FROM** news **WHERE** id_news = -1 **OR** 1=1

Таким образом, изменение входных параметров путём добавления в них конструкций языка SQL вызывает изменение в логике выполнения SQL-запроса (в данном примере вместо новости с заданным идентификатором будут выбраны все имеющиеся в базе новости, поскольку выражение 1=1 всегда истинно — вычисления происходят по кратчайшему контуру в схеме).

Внедрение в строковые параметры

Предположим, серверное ПО, получив запрос на поиск данных в новостях параметром search_text, использует его в следующем SQL-запросе (здесь параметры экранируются кавычками):

$search_text = $_REQUEST['search_text']; $res = mysql_query("SELECT id_news, news_date, news_caption, news_text, news_id_author FROM news WHERE news_caption LIKE('%**$search_text**%')");

Сделав запрос вида

http://example.org/script.php?search_text=Test мы получим

выполнение следующего SQL-запроса:

```
SELECT id_news, news_date, news_caption, news_text, news_id_author FROM news WHERE news_caption LIKE('%Test%')
```

Но, внедрив в параметр search_text символ кавычки (который используется в запросе), мы можем кардинально изменить поведение SQL-запроса.

Например, передав в качестве параметра search_text значение ')+and+(news_id_author='1, мы вызовем к выполнению запрос:

```
SELECT id_news, news_date, news_caption, news_text, news_id_author FROM news WHERE news_caption LIKE('%') and (news_id_author='1%')
```

Защита от атак типа внедрение SQL-кода

Для защиты от данного типа атак необходимо тщательно фильтровать входные параметры, значения которых будут использованы для построения SQL-запроса.

Фильтрация строковых параметров

Предположим, что код, генерирующий запрос (на языке программирования Паскаль), выглядит так:

```
statement := 'SELECT * FROM users WHERE name = ''' +
userName + ''';';
```

Чтобы внедрение кода (закрытие строки, начинающейся с кавычки, другой кавычкой до её завершения текущей закрывающей кавычкой для разделения запроса на две части) было невозможно, для некоторых СУБД, в том числе, для MySQL, требуется брать в кавычки все строковые параметры. В само́м параметре заменяют кавычки на \", апостроф на \', обратную косую черту на \\ (это называется «экранировать спецсимволы»). Это можно делать таким кодом:

```
statement := 'SELECT * FROM users WHERE name = ' +
QuoteParam(userName) + ';';
```

Для PHP фильтрация может быть такой:

```
<? $query = "SELECT * FROM users WHERE user='".mysql_real_escape_string($user)."';"; ?>
```

Фильтрация целочисленных параметров

Возьмём другой запрос:

```
statement := 'SELECT * FROM users WHERE id = ' + id + ';';
```

В данном случае поле **id** имеет числовой тип, и его чаще всего не берут в кавычки. Поэтому «закавычивание» и замена спецсимволов на escape-последовательности не проходит. В таком случае помогает проверка типа; если переменная **id** не является числом, запрос вообще не должен выполняться.

Например, на Delphi для противодействия таким инъекциям помогает код:

```
if TryStrToInt(id, id_int) then statement :=
Format('SELECT * FROM users WHERE id =%0:d;',
[id_int]);
```

Для PHP этот метод будет выглядеть так:

```
$query = 'SELECT * FROM users WHERE id = ' . (int)$id;
```

Усечение входных параметров

Для внесения изменений в логику выполнения SQL-запроса требуется внедрение достаточно длинных строк. Так, минимальная длина внедряемой строки в вышеприведённых примерах составляет 8 символов («**1 OR 1=1**»). Если максимальная длина корректного значения параметра невелика, то одним из методов защиты может быть максимальное усечение значений входных параметров.

Например, если известно, что поле **id** в вышеприведённых примерах может принимать значения не более 9999, можно «отрезать лишние» символы, оставив не более четырёх:

```
statement := 'SELECT * FROM users WHERE id = ' + LeftStr(id, 4) + ';';
```