ПРИМЕНИМЫЕ НА ПРАКТИКЕ СПОСОБЫ ПРЕДОТВРАЩЕНИЯ SQL-ИНЪЕКЦИЙ

Горячев Вячеслав Андреевич

студент, Самарский Национальный Исследовательский Университет имени академика С.П.Королёва РФ, г. Самара

Буторов Владислав Андреевич

студент, Самарский Национальный Исследовательский Университет имени академика С.П.Королёва РФ, г. Самара

Додонов Михаил Витальевич

научный руководитель, доц., Самарский Национальный Исследовательский Университет имени академика С.П.Королёва РФ, г. Самара

Проблемы безопасности SQL-баз данных

В современном обществе роль информации возрастает постоянно, и для удовлетворения потребностей этого самого общества разрабатываются многочисленные информационные системы, получающие, хранящие, обрабатывающие и распространяющие информацию. Для хранения информации используются базы данных – совокупность систематизированных данных, представленная в объектной форме, предоставляющей удобный доступ к данным. Значительную часть баз данных составляют базы на основе SQL – специализированного языка программирования для управления базами данных.

Современные информационные системы хранят в своих базах данных значительное количество личной информации пользователей, и именно поэтому вопрос безопасности баз данных очень актуален. И самой распространённой атакой на системы управления базами данных является SQL-инъекция.

SQL-инъекция

SQL-инъекция – атака, заключающаяся во внедрении злоумышленником вредоносного SQL-кода в SQL-запрос. Причиной возможности использовать внедрение вредоносного кода является недостаточная обработка введённых пользовательских данных. На примере Web-сайта рассмотрим принцип работы SQL-инъекций.

Рисунок 1. Блок-схема нормальной работы Web-сайта

При открытии пользователем страницы некоторого Web-сайта браузер выполняет запрос определенного вида на Web-сервер (рис. 1). Данный запрос содержит в себе параметры, при помощи которых Web-сервер определяет, какую именно информацию пользователь хочет получить; в данном случае таким параметром является уникальный идентификатор (id = 1991). На основе этих параметров Web-сервер формирует SQL-запрос к базе данных, результат которого будет возвращён пользователю.

Рисунок 2. Блок-схема исполнения вредоносного запроса на Web-сайте

Злоумышленник может изменить запрос, отправляемый на Web-сервер, внедрив в него особым образом вредоносный код. При недостаточной фильтрации входных параметров на Web-сервере вредоносный код станет частью SQL-запроса и будет исполнен системой управления базами данных. В результате выполнения вредоносного запроса злоумышленник сможет прочитать, модифицировать или уничтожить данные – совершить операцию, для которой у злоумышленника как пользователя недостаточно привилегий.

Классификация SQL-инъекций

Все SQL-инъекции можно классифицировать по типу передаваемого параметра, а также по способу эксплуатации. При рассмотрении вопроса защиты классификация по типу параметра играет более значимую роль.

Способ эксплуатации вредоносного запроса влияет как на сам внедряемый злоумышленником запрос, так и на результат. По данному критерию все SQL-инъекции можно разделить на 5 групп:

· UNION-based – содержащие в себе ключевое слово UNION, позволяющее объединить несколько запросов в один и получить общий вывод;

· Time-based – основанные на использовании преднамеренных задержек, выполняющихся при определенном условии. Отличительным моментом является то, что информация получается путем анализа длительности выполнения запроса;

· Error-based – использующие вывод ошибок систем управления базами данных как основной источник информации;

· Boolean-based (blind) – включающие в себя некоторое условие, выполнение которого не отразится на результате запроса, однако невыполнение, напротив, станет причиной пустого результата;

· Out-of-band (удаленный вывод) – выполняющие запрос, содержащий результат вывода, на удаленный сервер злоумышленника.

Классификация по типу параметра более простая, так как он бывает только двух видов:

· Числовой. SQL-запрос ожидает на вход число, но при недостаточной фильтрации вводимых данных может принять строчку, содержащую в себе вредоносную нагрузку. Поскольку числа имеют строго определённый формат, входные данные легко проверить: они могут содержать только цифры, точку и префиксный минус;

· Строковый. В SQL-запросе место для строкового параметра окружено кавычками. При недостаточной фильтрации злоумышленник может выйти за их границы и продолжить запрос. Чтобы этого не произошло, необходимо экранировать кавычки и всё, что может быть интерпретировано как кавычки, чтобы система управления базами данных не восприняла их как границу строкового параметра;

Описание и реализация алгоритма фильтрации

Как было написано выше, фильтрация числовых параметров сводится к проверке их на соответствие заданному формату. Эту проверку легко выполнить с использованием конечного автомата.

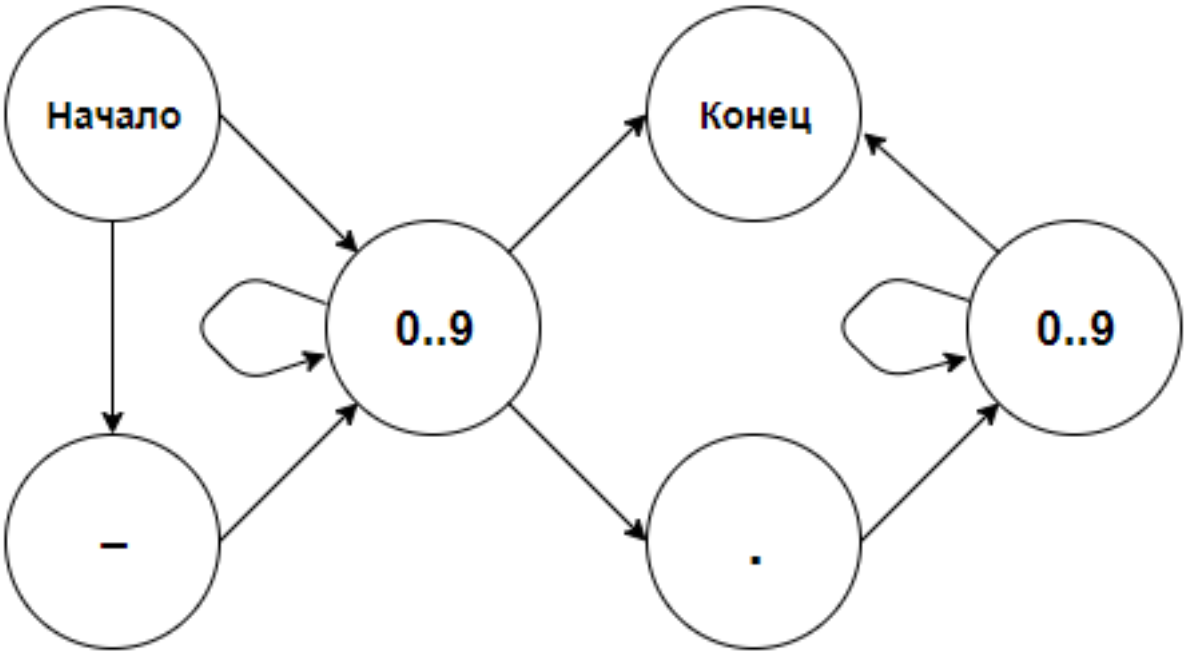

Рисунок 3. Конечный автомат для проверки числового параметра

Приведенный конечный автомат допускает наличие минуса в начале числа, а также дробной части в конце числа, и его реализация возможна с использованием регулярных выражений.

Для предотвращения внедрения кода в строковый параметр SQL-запроса нам необходимо экранировать кавычки и их кодированные аналоги. Тогда система управления базами данных не воспримет их как завершение вводимого параметра, и код злоумышленника не сможет выполниться.

Таблица 1.

Соответствие замен

Подстрока до замены	Подстрока после замены

‘	\’
“	\”
%22	\%22
%27	\%27
\x22	\\x22
\x27	\\x27
\u022	\\u022
\u027	\\u027

Если произвести замены представленным в таблице образом, вредоносный код, имея экранированные кавычки, не сможет покинуть пределы строки и останется неисполненным.

Вывод

В данной работе мы рассмотрели понятие, принцип работы и виды SQL-инъекций, объяснили, как их предотвратить, предложили алгоритм, способный защитить систему управления базами данных от вредоносного кода, а также реализовали его и привели в приложении исходный код алгоритма, написанный на языке высокого уровня Python.

Список литературы:

1. Смирнов С.Н. Безопасность систем баз данных. Гелиос АРВ, 2007. – 352с.

2. Дейт К. Дж. Введение в системы баз данных. Вильямс, 2016. – 1327с.

Приложение.

Код алгоритма

```
import re

PATTERN_NUMBER = r"^-?\d+(\.\d)?\d*$"

REPLACE_DICT = {
    "'": "\\'",
    '"': '\\"',
    "%22": "\\%22",
    "%27": "\\%27",
    "\\x22": "\\\\x22",
    "\\x27": "\\\\x27",
    "\\u022": "\\\\u022",
    "\\u027": "\\\\u027"
}

def check_numeric(input):
    return re.match(PATTERN_NUMBER, input) is not None


def process_string(input):
    for k,v in REPLACE_DICT.iteritems():
        input = input.replace(k,v)
    return input
```

Рисунок 4. Код алгоритма