Захист веб-додатків від SQL-ін’єкцій

Гліб Саєнко[1], Тетяна Гріненко[2]

1. Кафедра безпеки інформаційних технологій,
Харківський національний університет радіоелектроніки,
м. Харків, пр. Науки, 14,
E-mail: saenkogleb21@gmail.com

2. Кафедра безпеки інформаційних технологій,
Харківський національний університет радіоелектроніки,
м. Харків, пр. Науки, 14,
E-mail: tetiana.grinenko@nure.ua

The report examined the main types of SQL-injection and methods of protection against them.

Атака, база даних, веб-додаток, вразливість, SQL-ін’єкція.

I. Вступ

Потреби ринку інформаційних послуг нарівні з бурхливим розвитком телекомунікаційних технологій в останні десятиріччя стали вагомими причинами стрімкого розвитку технологій широкого обміну інформацією, побудови потужних територіальних і глобальних обчислювальних мереж. Такою мережею є Інтернет і найбільшу частину у ньому займають веб-додатки (веб-сайти). Існує величезна кількість різного роду вразливостей, за допомогою яких зловмисник може зламати веб-додаток і видалити або вкрасти чужу інформацію. Однією з таких уразливостей є SQL-ін’єкція – впровадження шкідливого SQL-коду в тіло HTTP-запиту до веб-додатку [1].

II. SQL-ІН’ЄКЦІЯ

SQL-ін’єкція – це один з поширених способів злому сайтів та програм, що працюють з базами даних, заснований на впровадженні в запит до веб-додатку довільного SQL-коду [1,2].

Атака типу впровадження SQL-коду, в залежності від типу системи управління базами даних та умов впровадження, дає можливість атакуючому виконати довільний запит до бази даних (наприклад, прочитати вміст будь-яких таблиць, видалити, змінити або додати дані), отримати можливість читання та/або запису локальних файлів та виконання довільних команд на сервері. Атака типу впровадження SQL-коду може бути можлива за некоректної обробки вхідних даних, що використовуються в SQL-запитах. Для безпеки веб-додатків рекомендується, щоб розробники ніколи не використовували введені користувачем дані безпосередньо для формування SQL-запитів до бази даних. Найкращим рішенням є, звичайно, повна ізоляція веб-додатків від створення SQL-запитів, тобто перенесення всього функціоналу, пов'язаного з їх формуванням, на рівень збережених процедур серверу баз даних.

III. Приклади SQL-ІН’ЄКЦІЙ

Припустимо, що серверне програмне забезпечення, отримавши вхідний параметр id, використовує його для створення SQL-запиту. Розглянемо такий PHP-скрипт [2]:

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

Якщо на сервер переданий параметр id, що дорівнює 5 (наприклад: http://example.org/script.php?id=5), то буде виконаний такий SQL-запит:

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

Але, якщо зловмисник передасть як параметр id рядок -1 OR 1=1 (наприклад, таким чином: http://example.org/script.php?id=-1+OR+1=1), то виконається запит:

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

Таким чином, зміна вхідних параметрів шляхом додавання в них конструкцій мови SQL викликає зміну в логіці виконання SQL-запиту (в цьому прикладі замість новини із заданим ідентифікатором будуть вибрані всі наявні в базі новини, оскільки вираз 1=1 завжди істинний).

Мова SQL дозволяє об'єднувати результати декількох запитів за допомогою оператора UNION. Це надає зловмисникові можливість отримати несанкціонований доступ до даних.

Розглянемо скрипт відображення новини (ідентифікатор новини, яку необхідно відобразити, передається в параметрі id):

```
$res = mysql_query("SELECT id_news, header, body, author FROM news WHERE id_news = ". $_REQUEST['id']).
```

Якщо зловмисник передасть як параметр id конструкцію -1 UNION SELECT 1,username, password,1 FROM admin, це викличе виконання SQL-запиту:

```
SELECT id_news, header, body, author FROM news WHERE id_news =-1 UNION SELECT 1,username,password,1 FROM admin.
```

Оскільки новини з ідентифікатором -1 завідомо не існує, з таблиці news не буде вибрано жодного запису, проте в результат потраплять записи, несанкціоновано відібрані з таблиці admin внаслідок ін'єкції SQL.

Розщеплення SQL-запиту.

Для розділення команд в мові SQL використовується символ ; (крапка з комою), включаючи цей символ до запиту, зловмисник отримує можливість виконати декілька команд в одному запиті. Наприклад, якщо в параметри скрипта

```
$id = $_REQUEST['id'];
$res = mysql_query("SELECT * FROM news WHERE id_news = $id");
```

зловмисником передається конструкція, що містить крапку з комою, наприклад, 12;INSERT INTO admin (username, password) VALUES ('HaCkEr', 'foo'); то в одному запиті будуть виконані 2 команди

```
SELECT * FROM news WHERE id_news = 12;
```

INSERT INTO admin (username, password) VALUES ('HaCkEr', 'foo');

і в таблицю admin буде несанкціоновано доданий запис HaCkEr.

IV. Атаки типу впровадження SQL-коду

Пошук скриптів, уразливих для атаки.

На цьому етапі зловмисник вивчає поведінку скриптів сервера при маніпуляції вхідними параметрами з метою виявлення їх аномальної поведінки. Маніпуляція відбувається всіма можливими параметрами [2,3]:

– даними, переданими через методи POST і GET;
– значеннями [HTTP-Cookie];
– HTTP_REFERER (для скриптів);
– AUTH_USER та AUTH_PASSWORD (при використанні автентифікації).

Як правило, маніпуляція зводиться до підстановки в параметри символу одинарної (рідше подвійної або зворотної) лапки.

Аномальною поведінкою вважається будь-яка поведінка, при якій сторінки, що одержані до і після підстановки лапок, розрізняються (і при цьому немає повідомлення «неправильний формат параметрів»).

Найчастіші приклади аномальної поведінки:

– виводиться повідомлення про різні помилки;
– при запиті даних, запитувані дані не виводяться взагалі, хоча сторінка відображається і т. д.

Слід враховувати, що відомі випадки, коли повідомлення про помилки, в силу специфіки розмітки сторінки, не видно в браузері, хоча і присутні в її HTML-коді.

V. Захист від атак типу впровадження SQL-коду

Фільтрація рядкових параметрів [2,3].

Припустимо, що код, який генерує запит (на мові програмування Паскаль), виглядає так:

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

Щоб впровадження коду було неможливо, для деяких систем управління базами даних, в тому числі, для MySQL, потрібно брати в лапки всі рядкові параметри. У самому параметрі замінюють лапки на \", апостроф на \', зворотну косу риску на \\ (це називається «екранувати спецсимволи»). Це можна робити, використовуючи такий код:

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

для PHP фільтрація може бути такою:

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

Фільтрація цілочисельних параметрів [2,3].

Візьмемо інший запит:

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

У цьому випадку поле id має числовий тип, і його найчастіше не беруть в лапки. У такому випадку допомагає перевірка – якщо змінна id не є числом, запит взагалі не повинен виконуватися.

Наприклад, на Delphi для протидії таким ін'єкціям використовується код:

```
id_int:= StrToInt(id);
statement:= 'SELECT * FROM users WHERE id = ' + IntToStr(id_int) + ';'.
```

У випадку помилки функція StrToInt викличе виняток класу EConvertError, і в його обробнику можна буде вивести повідомлення про помилку. Подвійне перетворення забезпечує коректну реакцію на числа у форматі $132AB (шістнадцяткова система числення). На стандартному Паскалі, який не вміє обробляти виняток, код дещо складніший.

Для PHP цей метод буде виглядати так:

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

Усікання вхідних параметрів [2,3].

Для внесення змін в логіку виконання SQL-запиту потрібно впровадження достатньо довгих рядків. Так, мінімальна довжина такого рядка у наведених вище прикладах становить 8 символів («1 OR 1=1»). Якщо максимальна довжина коректного значення параметра невелика, то одним з методів захисту може бути максимальне усікання значень вхідних параметрів.

Наприклад, якщо відомо, що поле id у вищенаведених прикладах може приймати значення не більше 9999, можна «відрізати зайві» символи, залишивши не більше чотирьох:

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

Висновок

Актуальність проблеми безпеки веб-додатків зростає з кожним днем. Більшість вразливостей, що існують на даний момент в цій сфері, пов'язані з помилками і недоліками, допущеними на етапі розробки сайту. В цій роботі були досліджені атаки типу SQL-ін’єкція. Наведені приклади реалізації, методи впровадження ін’єкції та основні методи захисту від впровадження SQL-коду. Метою подальших робіт є розробка атак типу SQL-ін’єкція на спеціалізовані веб-додатки та розробка методів захисту від них на мові програмування Golang.

Література

[1] Dafydd Stuttard, Marcus Pinto. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws 2 edition Wiley. 2011. 912c.

[2] Скембрейц Дж. Безопасность Web-приложсний – готовые решения / Дж. Сксмбрейц, М. Шема. – М.: Издательский дом «Вильямс», 2003. – 384 с.

[3] Habrahabr [Електронний ресурс] Методы обхода защитных средств веб-приложений при эксплуатации SQL-инъекций https://habr.com/ru/post/326362/ [дата звернення 06.10.2019].