Міністерство освіти та науки України

НТУУ «КПІ ім. Ігоря Сікорського»

Кафедра АПЕПС

Звіт з лабораторної роботи №2

З дисципліни «Методи захисту WEB-систем від основних видів атак»

Виконав студентка ТЕФ  
5 курсу гр.ТІ-71мп  
Ніколаєв В.В.

Перевірив: ас. Варава І. А.

Київ-2017

# Як боротись з SQL-ін’єкцією за допомогою PHP

SQL-ін’єкція це найбільш небезпечний тип атаки, адже саме вона стоїть за незліченною кількістю випадків злому які пережили корпоративні сайти та портали, і просто особисті домашні сторінки. Насправді захистити свій проект доволі легко – для цього потрібно перш за все зрозуміти в чому суть цієї проблеми та внести в свій код деякі зміни для захисту.

1. **Що таке SQL-ін’єкція**

SQL- ін’єкцією називають дії, які комусь потрібно здійснити, для того щоб виконати свій власний запит до бази даних, без вашого відома. Найчастіше це відбувається коли ви пропонуєте користувачу відправити якусь інформацію на сервер, але натомість бажаної інформації зловмисник відсилає свій запит, який, по необережності, виконується сервером. За допомогою такого запиту зловмисник може не тільки отримати заборонену інформацію з бази даних, але й, при певних умовах, внести до неї зміни, і навіть виконати системні команди.

1. **Приклади SQL-ін’єкцій**

Відправити свій запит зловмисник може не тільки вписавши його в поле для вводу інформації (використовуючи $\_POST ), він може також підставити свої $\_GET змінні до адресної стрічки, або власноруч змінити свої $\_COOKIE. Тому слід бути обачними при роботі з цими глобальними масивами даних.

2.1 **$\_POST:**

Якщо у вас на сторінці є форма для вводу певної інформації, зловмисник може використати її поля для своїх цілей. Замість очікуваної стрічки (ім’я, пароль, тощо) він введе в таке поле свій власний запит.

Більшість таких SQL-ін’єкцій виглядають наступним чином:

asd' or 1=1--

Скажімо ми маємо форму для авторизації користувачів. Якщо ми введемо такий код в поле логіну, ми зможемо використати нашу SQL-ін’єкцію для отримання доступу навіть без належних перевірок. Як воно працює? Розглянемо який все ж таки запит ми отримаємо в результаті наших дій:

SELECT \* FROM users WHERE username = 'asd' or 1=1--' and password = 'asd'

Отже як видно з прикладу, наш код буде успішно виконано. А так як вираз 1=1 буде завжди повертати true, ми гарантовано отримаємо доступ.

Вас швидше за все зацікавить, для чого нам подвійний дефіс (--). Цей подвійний прочерк в кінці стрічки повідомляє SQL серверу, що йому слід проігнорувати решту запиту. Якщо ж ви хочите написати ін’єкцію не для SQL серверу, то в такому випадку, вам у пригоді стане, заміна подвійного дефісу на одинарну кавичку.

Зверніть увагу, наведений вище приклад це просто найбільш стандартний варіант, але далеко не єдиний. Іх кількість просто вражає, і все залежить від того як працює голова зловмисника. Приклади ще деяких найбільш поширених ін.’єкцій:

') or ('1'='1

"or "1"="1

' or '1'='1

Or 1=1--

" or 1=1--

' or 1=1--

2.2 **$\_GET**

Також доволі часто зловмисники використовують адресну стрічку (URL) для своїх атак. Цей метод так як і попередній, не менш небезпечний. Коли на сервері використовується PHP та MySQL (на даний момент, найпопулярніша комбінація), адреса до скрипта як правило виглядає приблизно наступним чином:

http://somesite.com/login\_script.php?id=1

Додаючи до такої стрічки трішки SQL-я можна робити страшні речі:

http://somesite.com/login\_script.php?id=1‘; DROP TABLE login; #

В даному випадку використано знак # натомість подвійного дефісу, так як він говорить SQL серверу проігнорувати всі наступні запити, які йтимуть після нашого. І що саме небезпечніше (якщо ви ще не помітили), ми щойно сказали серверу вилучити табличку з користувачами. Даний приклад чудово демонструє наскільки небезпечними можуть бути SQL-ін’єкції.

1. **Що потрібно зробити для захисту своїх скриптів від SQL-ін’єкцій**

Ми уже знаємо, що вразливість на SQL-ін’єкції виникає тоді, коли інформація від користувачів потрапляє в запит, до бази даних, без відповідної обробки. Отже наступним кроком йде написання безпечних скриптів.

На щастя, дана небезпека відома вже досить давно. В PHP навіть з’явилась спеціальна функція (починаючи з версії 4.3.0), яка бориться з цим типом атак — mysql\_real\_escape\_string.

mysql\_real\_escape\_string перетворює стрічку на безпечну для вживання в запитах до бази даних, шляхом екранування всіх потенційно небезпечних символів. Як правило таким сиволом є одинарна кавичка ('), яка після використання цієї функції буде екранована (\').

Для того щоб захиститись від SQL-ін’єкції, всі зовнішні параметри ($\_GET, $\_POST, $\_COOKIE), слід перед тим як включити їх до SQL запиту опрацювати за допомогою функції mysql\_real\_escape\_string(), а в самому запиті помістити їх до одинарних кавичок. Якщо дотримуватись цього простого правила, тоді дії зловмисника приведуть до формування безпечних запитів, так як весь текст його SQL-ін’єкцій тепер всередині кавичок.

Давайте розглянемо що ж насправді виконуватиме сервер при такій обробці:

SQL-ін’єкція (стрічка яку зловмисник вписав в поле «Логін» натомість свого логіну):

sql' or 1=1--

PHP:

$name = mysql\_real\_escape\_string($\_POST['username']);

$res = mysql\_query("SELECT \* FROM users WHERE username = '".$name."' and password = 'asd'");

MySQL:

SELECT \* FROM users WHERE username = 'sql\' or 1=1--' and password = 'asd'

Тобто, при такому запиті, натомість небезпечних дій, ми намагаємось вибрати дані користувача в якого доволі дивний username (sql' or 1=1--)

Можна піти далі й написати функціонал який буде автоматично обробляти масиви $\_GET, $\_POST та $\_COOKIE відповідним чином, але це все залижить тільки від ваших побажань. Запам’ятати слід тільки те що, потрібно захистити ВСІ місця де в базу даних передаються дані від користувача.