УДК 004.056.5
Зайко Т.А.[1], Лизя Є.С.[2]
[1]канд. техн. наук, доц. НУ «Запорізька Політехніка»
[2]студ. гр. КНТ-217 НУ «Запорізька політехніка»

ЗАХИСТ ВІД SQL-ІН'ЄКЦІЙ В PHP І MYSQL

Вебсайти стали частиною нашого життя. Більшість сьогоденних справ ми виконуємо за допомогою функціоналу тих чи інших веб сервісів, а це говорить про те, що веб-сайти мають доступ до баз даних, де зберігається безліч особистої інформації користувачів. Хакери та інші злочинники використовують, так звану, SQL-ін'єкцію для того, щоб вкрасти та змінити інформацію, яка зберігається в БД (базі даних). Саме тому виникла необхідність створити методи для захисту інформації від SQL-ін'єкцій.

SQL-ін'єкція [1] – це атака, спрямована на веб-додаток, в ході якої конструюється SQL-вираз з введених користувачем даних, шляхом простої конкатенації (наприклад, $ query = "SELECT * FROM users WHERE id =". $ _ REQUEST ["id"]) . У разі успіху хакер може змінити логіку виконання SQL-запиту так, як це йому потрібно. Найчастіше він виконує простий fingerprinting СУБД (визначення типу системи управління базами даних), а також витягує таблиці з найбільш "цікавими" іменами (наприклад "users"). Після цього, в залежності від привілеїв, з якими запущено вразливий додаток, він може звернутися до захищених частин бекенду (серверна сторона) веб-додатку (наприклад, прочитати файли на стороні адміністратора або виконати довільні команди).

Для захисту вебсайтів розробники винайшли багато рішень. Нище наведені два методи [2], які є найбільш популярними та ефективними серед вебпрограмістів.

Перший метод полягає в тому, що дані підставляються в запит тільки через плейсхолдери (заповнювачі). Будь-які дані повинні потрапляти в запит не безпосередньо, а через якогось представника, а саме – підставний вираз.

Найкращим варіантом буде форматувати дані безпосередньо перед виконанням запиту – таким чином ми завжди будемо впевнені в тому, що дані форматуються правильно, це робиться тільки один раз, і відформатовані дані потраплять строго за призначенням – в БД і нікуди більше. Існує два варіанти реалізації плейсхолдерів – серверний і клієнтський. У першому випадку запит так і йде на сервер з плейсхолдерами, а дані відправляються окремо від нього. Англійською має назву native prepared statements – «рідні» підготовлені вирази – тобто, обробка плейсхолдерів здійснюється самою СУБД (системою управління базами даних) на сервері. У другому випадку дані форматуються і підставляються в рядок запиту на місце плейсхолдерів прямо на клієнті, формуючи класичний SQL запит, який потім йде в базу звичайним порядком.

Другий метод полягає в тому, щоб ідентифікатори і ключові слова підставлялись тільки з білого списку, прописаного в нашому коді. Ми зіштовхуємося з необхідністю підставляти в запит не тільки дані, але і інші елементи – ідентифікатори (імена полів і таблиць) і навіть елементи синтаксису, ключові слова. Нехай навіть такі незначні, як DESC або AND, але вимоги до безпеки таких підстановок все одно повинні бути не менш суворими. Суть методу полягає в тому, що всі можливі варіанти вибору повинні бути жорстко прописані в коді, і в запит повинні потрапляти тільки вони, на підставі призначеного для користувача введення.

Я вважаю, що для повноцінного захисту вебсайту від SQL-ін’єкції потрібно використовувати обидва методи, оскільки лише одного з них може бути не достатньо. Оскільки, якщо підставляти імена полів без попередньої фільтрації (тобто використовувати тільки метод з використанням плейсхолдерів), можна отримати ін'єкцію іншого роду – адже користувач тоді може вписати в ті імена полів, які йому змінювати не можна. Скажімо, якщо ми формуємо SQL (Structured Query Language) запит автоматично на базі масиву $ _POST, то хакер при реєстрації додає в форму поле admin зі значенням «1» і стає адміністратором. Отже, на мою думку, спочатку потрібно отримувати ідентифікатор з білого списку, а потім додавати його через плейсхолдер.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. SQL инъекции [Електрон. ресурс]. – Режим доступу: https://xakep.ru/2011/12/06/57950/.

2. Защита от SQL инъекций [Електрон. ресурс]. – Режим доступу: https://habr.com/ru/post/148701.