ДЕКІЛКА ПРИЧИН КРАДІЖОК ДАНИХ

Лебединський М.О. , студент, maksim.lebedinskiy@openmailbox.org;
ДонНТУ, Покровськ, Україна

На сьогоднішній день крадіжка даних є дуже актуальною. Зараз дуже багато веб-ресурсів не можуть захистити себе. Головною причиною цього є нездатність та лінь розробників та адміністраторів, які допускають дуже великі помилки. Прикладом цього є SQL-injection. Також однією із причин є сам користувач, який створює дуже прості паролі та дуже часто використовує їх на декількох ресурсах.

Для початку потрібно зрозуміти що таке SQL. Це непроцедурна мова програмування, яка застосовується для створення модифікацій та управління даними в реляційній базі даних, яка керується відповідною системою управління базами даних.

Також потрібно зрозуміти, що таке SQL-injection, та чим саме вони небезпечні. SQL-injection це один із розповсюджених способів злому сайтів та програм, які працюють з базами даних. Ця атака базується на впровадженні у запит випадкового SQL-кода. Це впровадження виконується та може дати можливість зловмиснику виконати свою команду, яку не планував розробник ресурсу.

Найпростіший спосіб проникнення до ресурсу, це додавання свого коду до форм введення. Наприклад: *http://www.sqlinjection.com/index.php?pic=1'* Цей код змінює запит. І видасть помилку, якщо параметр "pic" не фільтрується на символ ('). Ми побачимо повідомлення від бази даних, у якому буде вказано у чому суть помилки. Найкраще не демонструвати це повідомлення користувачу, так як у повідомленні може знаходитися важлива та конфіденційна інформація.

Видів ін'єкцій дуже багато, але зосередимося на їх виправленні. Виправленням SQL injection є фільтрація вхідних даних. Фільтрувати потрібно одинарні та подвійні лапки, крапки з комами, та символи для коментарів, а саме подвійне тире. Але потрібно приймати до уваги, що користувач може використовувати ці символи у своєму паролі, тому його потрібно сповістити, що деякі символи недопустимі і їх не потрібно використовувати.

Головне, це перевіряти данні на 3 рівнях:

1) на рівні користувача (але це не зовсім безпечно, так як є можливість послати запит в обхід функції перевірки);
2) на рівні сервера;
3) на рівні бази даних.

Сформуємо принципи, які допоможуть захиститися від ін'єкцій:

1) Перший варіант: сформувати білий або чорний список команд, які можна використовувати. Цей процес довгий та не завжди найкращий.

2) Якщо говорити о PHP. Обробляти данні функцією: *mysql_real_escape_string();*
ця функція екранує спецсимволи знаком \, що не дає виконуватися цим символам. Важливо: додані знаки екранування не йдуть в саму базу. При потраплянні в БД вони відкидаються.
3) Потрібно не забувати брати данні в ' ';
Наприклад: *SELECT * FROM table WHERE name = 'Bill'*
4) Імена полів та таблиць потрібно брати в зворотні одинарні лапки (`);
Наприклад: *SELECT * FROM `table` WHERE `name` = 'Bill'*
5) Потрібно пам'ятати, що екранувати потрібно не лише данні, які йдуть від користувача, а усі данні які є в запиті.
6) Але, в той же час, потрібно екранувати лише те що потрібно, наприклад данні які будуть зображуватися у користувача не повинні містити в собі зайві спеціальні символи екранування.
7) Якщо запит динамічно формується, то не можна вставляти оператори, імена БД, таблиці, полів в запит напряму. Найкраще усі варіанти заздалегідь прописувати у скрипті.
Наприклад:

```
$order = array("name", "price", "qty");
$key=array_search($_GET['sort'],$orders);
$orderby=$orders[$key];
$query="SELECT * FROM `table` ORDER BY $orderby";
```

8) Використовувати параметризовані запроси. Де змінні частини замінюються маркерами, наприклад знак питання (?).

Багато спеціалістів рекомендують шифрувати данні в БД. Це допомагає надійно захистити свою інформацію. Про це правило також не потрібно забувати.

Також часто для усунення ін'єкцій використовують не звичайні запити, а запити загорнуті у функції, що дозволяє уникнути деяких варіантів ін'єкцій.

В місцях де можливо буде проходити текст, використовують подвійне перетворення в base64 і назад, тоді функція сприймає усі спеціальні символи (",',; та інше) як частину тексту і не реагує на них як на елемент синтаксису.

Прикладом цього коду є:

```
convert_from(decode(encode($5::bytea,'base64'),'base64')'UTF-8')
```

Де, $5 це порядковий номер параметра функції.

Також, у скрипті, який виконує запит необхідно використовувати попередню перевірку параметрів регулярним виразом.

Тому, найефективнішим захистом від SQL-injection є перевірка даних які вводить користувач. Перевірку необхідно здійснювати на стороні користувача, сервера та БД. Також для перевірки та виправлення вразливостей використовують спеціальні програми, такі як SQLmap.

SQL-injection дуже небезпечна вразливість, але навіть якщо БД буде викрадена, паролі ще потрібно розшифрувати. А для того, щоб збільшити час розшифрування необхідно вигадувати складні паролі. Сформуємо, який пароль є складним:

1) Він повинен бути більше 8 символів, а краще більше 10.
2) Він повинен містити в собі маленькі, великі літери, цифри, а також спецсимволи, якщо вони дозволені (@,#,!,$,(,) та інші).
3) Існує багато таблиць, які містять у собі хєш-функції слів та часто зустрічних паролів. Тому не можна використовувати у своїх паролях слова із словника чи видозмінені слова, це не дуже надійно.
4) Не можна використовувати імена комп'ютерів чи облікових записів.
5) І найголовніше правило, не можна використовувати один і той-же пароль на декількох ресурсах.
6) Також, рекомендують змінювати свої паролі хоча б раз у три місяці.

Усі формули будуть знаходитись у додатку. Це опис складності пароля у залежності від розміру і кількості використовуваних символів. Та розрахунки захисту від ін'єкцій.

Література

1. SQL Tutorial (2016) http://www.w3schools.com/sql/ (25 листопада 2016);
2. SQL Injection (2016) https://technet.microsoft.com/en- us/library/ms161953(v=sql.105).aspx
3. Introduction to Cybersecurity https://363725903.netacad.com/courses/418365
4. An SQL Injection can destroy your database. http://www.w3schools.com/Sql/sql_injection.asp
5. How can I prevent SQL injection in PHP? (1 October 2016) https://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php (25 листопада 2016)
6. Защита от SQL-инъекций http://phpfaq.ru/mysql/slashes (25 листопада 2016)

Анотація

Продемонстрована SQL-injection, суть проблеми, та декілька методів виправлення цієї вразливості. Було сформовано основні принципи забезпечення захисту від цієї вразливості. Було продемонстровано основні рекомендації, щодо правильного та безпечного пароля. Було сформовано рівні безпеки, на яких потрібно впровадити захист.

Ключові слова: SQL-injection, безпека, пароль, захист, база даних, вразливість, рекомендації.

Аннотация

Продемонстрирована SQL-injection, суть проблемы и несколько методов исправления этой уязвимости. Были сформированы основные принципы обеспечения защиты от этой уязвимости. Было продемонстрировано основные рекомендации по правильному и безопасному паролю. Были сформированы уровни безопасности, на которых нужно внедрить защиту.

Ключевые слова: SQL-injection, безопасность, пароль, защита, база данных, уязвимость, рекомендации.

Abstract

Demonstrated SQL-injection, the problem and several methods to correct this vulnerability. It was formed the basic principles of protection from this vulnerability. It has been demonstrated basic guidelines for proper and safe password. Safety levels have been formed, which is necessary to implement protection.

Keywords: SQL-injection, security, password, protection, database, vulnerability, recommendations.