МИНОБРНАУКИ РОССИИ
федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»
(Самарский университет)

Институт информатики, математики и электроники
Факультет механико-математический
Кафедра безопасности информационных систем
Дисциплина Модели безопасности компьютерных систем

Лабораторная работа №2

Тема: Ознакомление и использование SQLMap

Выполнили:
студенты 4 курса, группа 6442-100501D
ФИО:
Стрыгина В.Э.
Молостов О.А.
Круталев И.В.
Проверил: Бурлаков М.Е.

Работа проверена
« »_____202__г.
Оценка _____
Преподаватель_____

Самара 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Учитывая популярность SQLMAP и подобных ей инструментов, а также большое количество сайтов, подверженных SQL-инъекциям, необходимо знать принцип работы SQL-инъекций, а также методы защиты.

Для выявления, подвержен ли скрипт уязвимости, будет использован SQLMap. SQLMap — это инструмент с открытым исходным кодом для тестирования на проникновение, который автоматизирует процесс выявления и эксплуатации уязвимости SQL-инъекция и захват серверов баз данных. Имеет широкий набор возможностей, начиная от сбора отпечатков баз данных по полученной от них данным, до доступа к файловой системе и выполнения команд в операционной системе посредством внеполосных (out-of-band) подключений [1].

Целью данной работы является ознакомление со SQLMap.

Для достижения поставленной цели нами были поставлены следующие задачи:

1. написать скрипт для дальнейшего исследования;
2. осуществить атаки с помощью SQLMap;
3. ознакомиться с методами защиты от SQL-инъекций.

1 Написание скрипта

В ходе выполнения данной работы был запущен веб-сервер apache2. Проверить корректность его запуска можно с помощью браузера, зайдя на localhost (ip 127.0.0.1). Все файлы для веб-сервера apache2 хранятся в папке: /var/www/html

Создается отдельная папка. В ней создается PHP-скрипт подключения к ранее созданной однотабличной MySQL БД и использования ее ресурсов (рис. 1-2).

```
MariaDB [shop]> select* from users;
+----------+----------+------------+
| name     | password | cardnumber |
+----------+----------+------------+
| admin    | 112345   | no         |
| userTest | 1123456  | 1          |
| user     | 123      | kalc       |
+----------+----------+------------+
3 rows in set (0.000 sec)
```

Рисунок 1. Созданная БД

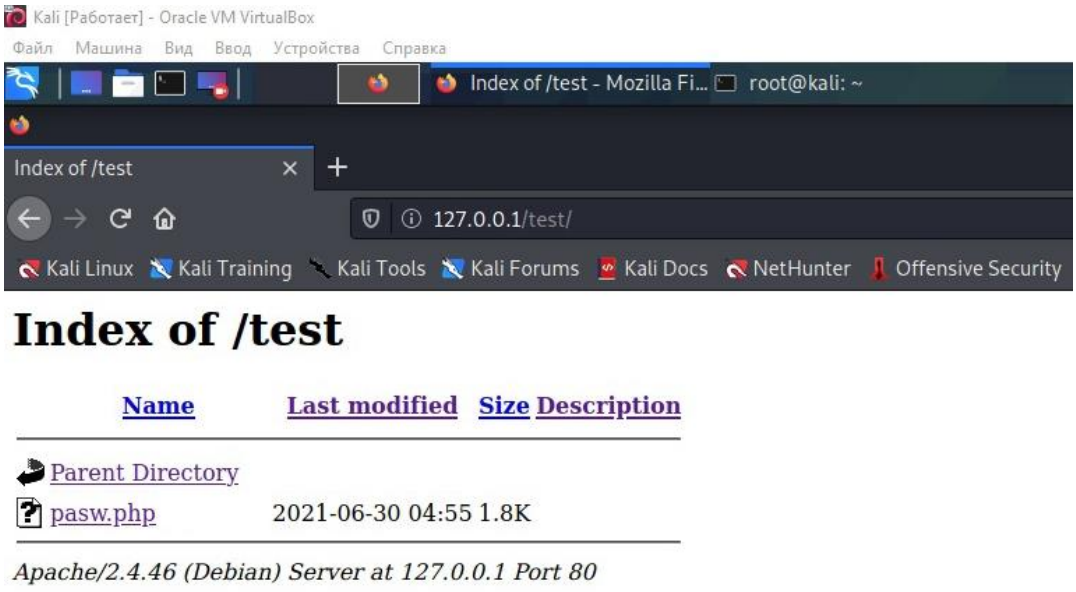

Рисунок 2. Созданный PHP-скрипт

На рисунке 3 представлено содержимое PHP-скрипта. Данный скрипт подключается к базе данных SHOP и по введенному логину и паролю находит запись в таблице users и выводит информацию из cardnumber (выводит номер карты введенного пользователя) (рис.4).

```php
<?php
  $servername = "localhost";  //replace your servername
  $username = "root";   //replace your username
  $password = "toor";        //replace your password
  $dbname = "shop";    //replace your database name

  // Create connection
   $conn = mysqli_connect($servername, $username, $password, $dbname);


    if ( isset($_GET["user"]) and isset($_GET["password"]) )
    {
        $username = $_GET["user"];
        $password = md5( $_GET["password"] );

        $val = "SELECT * FROM users WHERE name = '$username' and password = '$password'";
    $test = mysqli_query( $conn, $val );

        if ( !$res = mysqli_query( $conn, $val ) )
        {
            echo "Error: Our query failed to execute and here is why: <br/>";
            echo "Query: " . $val . "<br/>";
            echo "Errno: " . $conn->errno . "<br/>";
            echo "Error: " . $conn->error . "<br/>";
        }
        else
        {
            $data = $res->fetch_all();
        if ( $data )
                echo "Your card number: " . $data[0][2];
        else
        echo "Login error!";
}
    }
?>

<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Shop login</title>
    </head>
    <body>
        <form action="" method="GET" class="form-signin">
            <h2>Please login</h2>
            <label for="user" class="sr-only">Username</label>
            <input type="text" name="user" id="user" class="form-control" placeholder="Username" required="" autofocus=""><br/><br/>
            <label for="password" class="sr-only">Password</label>
            <input type="password" id="password" name="password" class="form-control" placeholder="Password" required=""><br/><br/>
            <button class="btn btn-lg btn-primary btn-block" type="submit">Log in</button>
        </form>
    </body>
</html>
```

Рисунок 3. PHP-скрипт

Рисунок 4. Результат работы скрипта

2 Проведение атак с помощью SQLMap

SQLMap — это инструмент с открытым исходным кодом для тестирования на проникновение, который автоматизирует процесс выявления и эксплуатации уязвимости SQL-инъекция и захват серверов баз данных. Имеет широкий набор возможностей, начиная от сбора отпечатков баз данных по полученной от них данным, до доступа к файловой системе и выполнения команд в операционной системе посредством внеполосных (out-of-band) подключений [1].

SQL-инъекция — это атака, направленная на веб-приложение, в ходе которой конструируется SQL-выражение из пользовательского ввода путем простой конкантенации (например, $query="SELECT * FROM users WHERE id=".$_REQUEST["id"]). В случае успеха атакующий может изменить логику выполнения SQL-запроса так, как это ему нужно. Чаще всего он выполняется простой fingerprinting СУБД, а также извлекаются таблицы с наиболее "интересными" именами (например "users"). После этого, в зависимости от привилегий, с которыми запущено уязвимое приложение, можно обратиться к защищенным частям бэк-энда веб-приложения (например, прочитать файлы на стороне хоста или выполнить произвольные команды).

Если сайт уязвим к SQL-инъекции, то возможно получать информацию из базы данных, в том числе дамп (всю) базу данных; изменять и удалять информацию из базы данных [2].

Есть пять основных классов SQL-инъекций, и все их поддерживает sqlmap [3]:

UNION query SQL injection. Классический вариант внедрения SQL-кода, когда в уязвимый параметр передается выражение, начинающееся с "UNION ALL SELECT". Эта техника работает, когда веб-приложения напрямую возвращают результат вывода команды SELECT на страницу: с использованием

цикла for или похожим способом, так что каждая запись полученной из БД выборки последовательно выводится на страницу. Sqlmap может также эксплуатировать ситуацию, когда возвращается только первая запись из выборки (Partial UNION query SQL injection).

Error-based SQL injection. В случае этой атаки сканер заменяет или добавляет в уязвимый параметр синтаксически неправильное выражение, после чего парсит HTTP-ответ (заголовки и тело) в поиске ошибок DBMS, в которых содержалась бы заранее известная инъецированная последовательность символов и где-то "рядом" вывод на интересующий нас подзапрос. Эта техника работает только тогда, когда веб-приложение по каким-то причинам (чаще всего в целях отладки) раскрывает ошибки DBMS.

Stacked queries SQL injection. Сканер проверяет, поддерживает ли веб-приложение последовательные запросы, и, если они выполняются, добавляет в уязвимый параметр HTTP-запроса точку с запятой (;) и следом внедряемый SQL-запрос. Этот прием в основном используется для внедрения SQL-команд, отличных от SELECT, например для манипуляции данными (с помощью INSERT или DELETE). Примечательно, что техника потенциально может привести к возможности чтения/записи из файловой системы, а также выполнению команд в ОС. Правда, в зависимости от используемой в качестве бэк-энда системы управления базами данных, а также пользовательских привилегий.

Boolean-based blind SQL injection. Реализация так называемой слепой инъекции: данные из БД в "чистом" виде уязвимым веб-приложением нигде не возвращаются. Прием также называется дедуктивным. Sqlmap добавляет в уязвимый параметр HTTP-запроса синтаксически правильно составленное выражение, содержащее подзапрос SELECT (или любую другую команду для получения выборки из базы данных). Для каждого полученного HTTP-ответа выполняется сравнение headers/body страницы с ответом на изначальный запрос

— таким образом, утилита может символ за символом определить вывод внедренного SQL-выражения. В качестве альтернативы пользователь может предоставить строку или регулярное выражение для определения "true"-страниц (отсюда и название атаки). Алгоритм бинарного поиска, реализованный в sqlmap для выполнения этой техники, способен извлечь каждый символ вывода максимум семью HTTP-запросами. В том случае, когда вывод состоит не только из обычных символов, сканер подстраивает алгоритм для работы с более широким диапазоном символов (например для unicode'a).

Time-based blind SQL injection. Полностью слепая инъекция. Точно так же как и в предыдущем случае, сканер "играет" с уязвимым параметром. Но в этом случае добавляет подзапрос, который приводит к паузе работы DBMS на определенное количество секунд (например, с помощью команд SLEEP() или BENCHMARK()). Используя эту особенность, сканер может посимвольно извлечь данные из БД, сравнивая время ответа на оригинальный запрос и на запрос с внедренным кодом. Здесь также используется алгоритм двоичного поиска. Кроме того, применяется специальный метод для верификации данных, чтобы уменьшить вероятность неправильного извлечения символа из-за нестабильного соединения.

Используя инструмент SQLMap, начинаем тестировать скрипт с помощью команды:

sqlmap —url="http://127.0.0.1/test/pasw.php?user=user&password=123". В результате работы данной команды были найдены уязвимости и проведены инъекции (рис. 5).

Для наглядности можно применить команду:

sqlmap —url="http://127.0.0.1/test/pasw.php?user=user&password=123" -D shop T users -C password —dump , чтобы вывести столбец password из базы данных shop (рис.6).

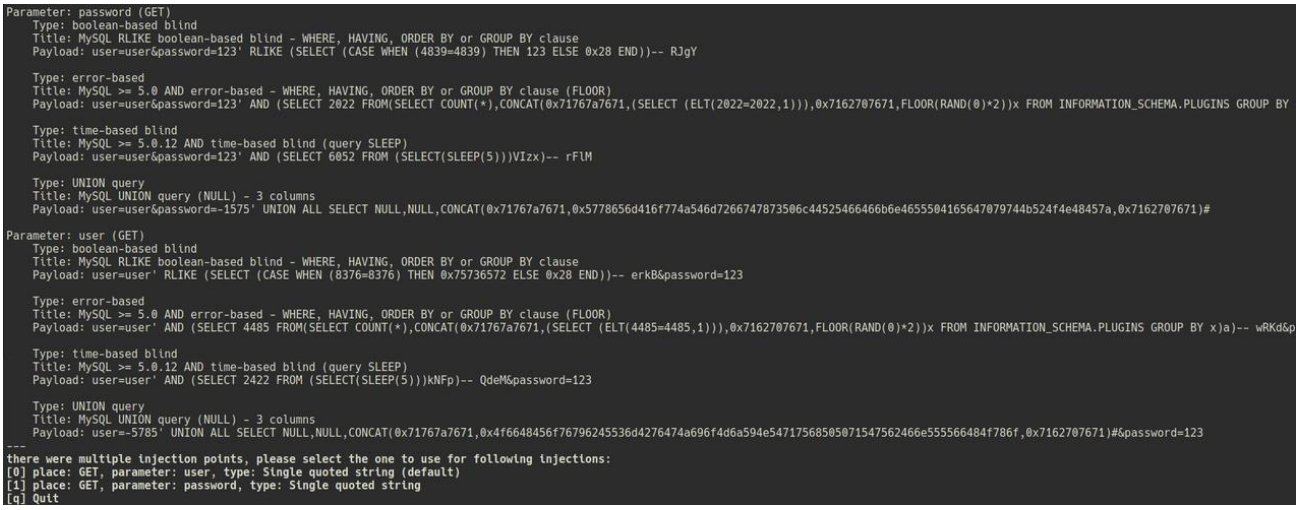

Рисунок 5. Результат работы sqlmap команды

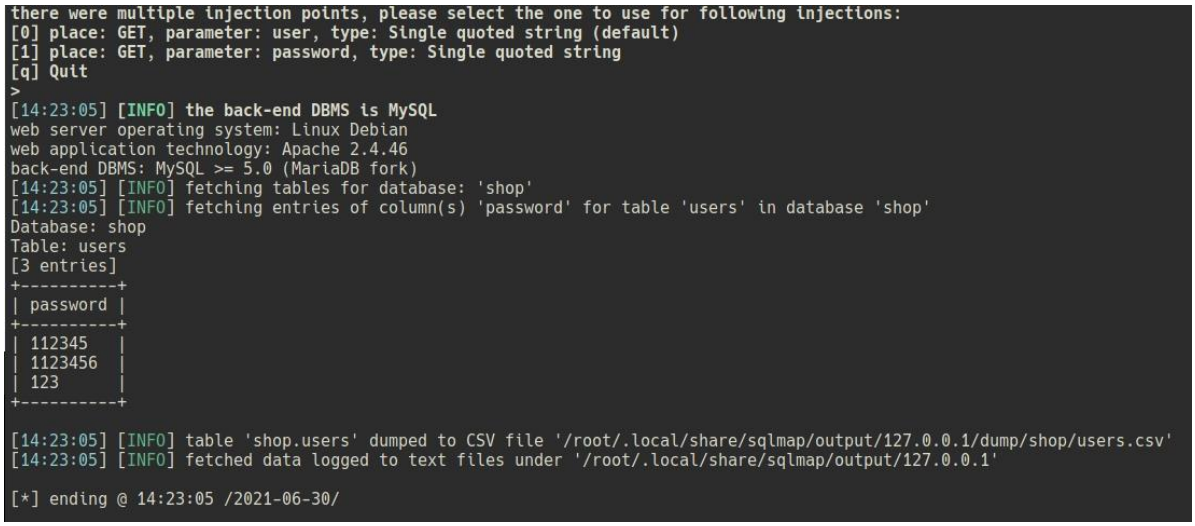

Рисунок 6. Вывод столбца password

С помощью команды sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=2" —dbs —tor —tor-type=SOCKS4 —tor-port=9150 —random-agent был проведен анализ стороннего ресурса с использованием tor-port. В результате были найдены уязвимости, проведены инъекции и выведены найденные базы данных (рис. 7).

Рисунок 7. Анализ стороннего ресурса

3 Методы защиты от SQL-инъекций

К методам защиты от SQL-инъекций относят:

1. Экранирование спец символов. Для mysqli функция экранирования называется real_escape_string. Экранировать (как и фильтровать) нужно абсолютно все данные, приходящие "с той стороны", т.е. от пользователя. Даже куки, даже те, которые, казалось бы, жёстко прописаны: например, в выпадающем списке три опции – и выбранная опция отправляется на сервер. Современные браузеры, даже без специальных плагинов, позволяют редактировать HTML-код на лету. Т.е. вы ожидаете одно из трёх значений, а получаете хитро сконструированный SQL-запрос [4];

2. Не пользоваться методом GET в формах. Передавать переменные этим способом очень опасно, потому что они оказываются на виду у пользователей. Если информация важная, то лучше использовать POST. Из ссылки злоумышленник может узнать не только имена переменных, но и какие значения должны быть в них — так он сможет выбрать оптимальную переменную для ввода инъекции. Лучше не использовать переменные прямиком из супермассива — лучше поместить их в другую переменную, предварительно проверив данные.

3. Проверка всех переменных на возможность инъекции [4]

Переменные, про которые часто забывают:

- скрытые поля в формах, которые пользователь, по вашему мнению, не видит и не может отредактировать;
- части адреса страницы;
- имена пользователей;
- данные передаваемые методом POST;
- данные из кукиз;
- данные из referer и т.д.

4. Не выводить сообщения об ошибках. Ошибки могут появляться в процессе запроса к базе данных, а также уже на следующих этапах — в процессе обработки полученных значений. Например, был возвращён пустой результат, а скрипт без проверки, сразу пытается его обработать, то появится ошибка. Если ваши скрипты продолжают оставаться уязвимыми, то отсутствие сообщений об ошибках даст возможность использовать только слепую SQL-инъекцию. Слепая инъекция — это тоже плохо, но это немного усложнит нападающему его задачу. Если SQL-инъекции нет, а сообщения об ошибках выводятся, то это может дать информацию о файлах, о структуре скриптов. Лучше всего, проверяйте и, если данные после очистки отсутствуют или после запроса базы данных ничего не получено, то отображайте главную страницу [4];

5. Проверять, откуда пришли данные. Недостаточно просто обработать данные — нужно узнать, откуда они пришли. Отследить источник можно несколькими способами:

- проверять его в $_SERVER[’HTTP_REFERER’];
- создать скрытое поле в форме;
- указать имя формы;
- указать имя кнопки отправки и так далее.

Данный метод отсеивает часть неопытных взломщиков, которые бросят свои потуги после нескольких попыток [5].

6. PDO. С помощью PDO и плейсхолдеров можно значительно снизить риск инъекции, потому что данные и запрос отправляются отдельно. Сначала производится подключение к базе, потом подготавливается запрос, затем отдельно указываются переменные, и, наконец, запрос выполняется. Данные отправляются уже в виде переменных. То есть если бы в переменной кто-то поставил лишнюю кавычку, сервер бы не подумал, что она — часть запроса. Поэтому попытка испортить запрос через переменную не сработала бы [5].

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы мы ознакомились со SQLMap, осуществили атаки на созданный нами скрипт, подключенный к базе данных MySQL, и проанализировали методы защиты от SQL-инъекций, благодаря которым пришли к выводу, чем больше защитных механизмов будут установлены, тем сохраннее будет информация.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SQLMap [Электронный ресурс] – URL: https://kali.tools/?p=816 (дата обращения: 30.06.2021)

2. Интсрукция по использованию SQLMap [Электронный ресурс] – URL: https://hackware.ru/?p=1928 (дата обращения: 30.06.2021)

3. SQLMap [Электронный ресурс] – URL: https://xakep.ru/2011/12/06/57950/ (дата обращения: 30.06.2021)

4. Защита сайта от взлома: предотвращение SQL-инъекций [Электронный ресурс] – URL: https://codeby.net/blogs/zashhita-sajta-ot-vzloma-predotvrashhenie-sql-inekcij/ (дата обращения: 30.06.2021)

5. Методы защиты от SQL-инъекции [Электронный ресурс] – URL: https://skillbox.ru/media/code/kak_zashchitit_sayt_ot_sql_inektsii/ (дата обращения: 30.06.2021)