УДК [004.56+003.26] (075.8)

Студ. А.Т. Черенков
Научн. рук. проф. П.П. Урбанович
(кафедра информационных систем и технологий, БГТУ)

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ ВЕБ-СЕРВИСА ПО ПАССАЖИРСКИМ ПЕРЕВОЗКАМ

Развитие информационных технологий обостряют проблему защиты данных [1, 2]. При разработке проекта сервиса по бронированию поездок была реализована защита от несанкционированного доступа к данным. Проект подобного сервиса подразумевает концепцию разделения уровней доступа к информации. Предусмотрено 3 уровня доступа к данным: клиент, водитель, оператор. Каждый из уровней имеет ограничения. Так клиент может видеть только свои данные и информацию о маршруте. Водитель может видеть все данные пассажиров в своём транспорте, а оператор имеет доступ ко всем данным компании.

Для получения доступа к базам данных проекта по умолчанию используется стандартный логин и пароль. Для повышения уровня защиты логин и пароль был изменен на набор строк и символов.

Второй шаг – анализ SQL-запросов. В PHP реализовано несколько способов работы с SQL-запросами. В проекте используется простейший вариант, реализация которого заключается в создании готового запроса из строки с подстановкой данных непосредственно в строку. Это самая большая проблема, которая была обнаружена во всем проекте и подлежала глобальному изменению.

Проблема заключается в ситуации, когда пользователь имеет доступ к содержимому переменных. А именно: переменные никак не валидируются на предмет стороннего кода. Способ доступа к данным при помощи передачи в переменные запроса целых конструкций из запросов, определяется как SQL-Injection [3]. При анализе была отработана атака при помощи строки URL в браузере. В проекте имеется передача ID маршрута при помощи URL строки:

http://autoslava/pages/php/seats.php?id=1 .

В примере пользователь получает доступ к маршруту под ID = 1. В случае, если регистрация открыта, страница отобразится корректно. Если данного ID не существует, страница будет пустая. ID – это переменная, значение которой передается при помощи URL. Если передать, например, ID=-1, появится пустая страница, что не предусмотрено проектом (рис. 1). Целью атаки является получение данных пассажиров из таблицы, которая достижима при продолжительных попытках взлома и хранит данные пассажира и его поездки.

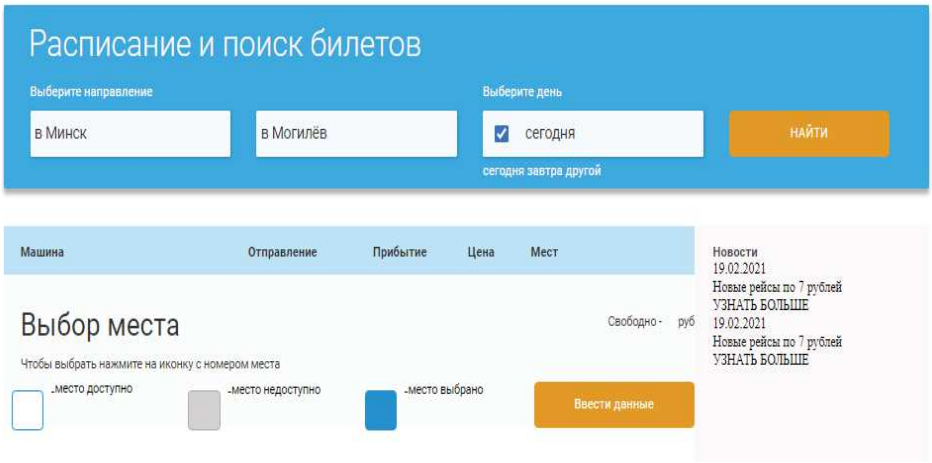

Рисунок 1 – Неверное отображение страницы при изменении входных данных

При использовании конструкции UNION есть возможность произвести объединение таблиц по ключу и получить данные из таблицы. При помощи следующего запроса можно получить информацию из результата такого объединения таблиц:

*autoslava/pages/php/seats.php?id=99 union (select * fromseats) -- .*

При исполнении сервером подобного запроса в переменную ID будет отправлена строка «*99 union(select * fromseats*) –». Переменная ID является последней в запросе, поэтому после передачи значения команда расширится при помощи union. Таким образом, злоумышленник получит доступ к любым данным о бронировании пользователями этой поездки.

Эта атака показала, что нет проверки вводимых данных пользователем в поля для ввода или переменные. Для реализации защиты необходимо устранить эту проблему. Для решения этой проблемы в PHP [4] существует несколько способов, которые позволяют на старте устранить любые попытки ввода неподходящих типов данных. Для начала стоит реализовать 1-й этап защиты, подобный черному списку (пример кода – в листинге 1). Это проверка вводимых данных на предмет «опасных» слов или значений, таких как UNION, LIKE, одинарные кавычки, --, эти символы нигде не вводятся в проекте, но таки данные может захотеть ввести пользователь. Это защитит сразу также от еще одной проблемы: создание файлов на сервере. При помощи команды имеется возможность создавать файлы без ограничений на сервере.

Второй шаг – это обработка запроса в виде строки с плейсхолдерами. Это защищает от расширения строки сторонними конструкциями. В листинге 2 приведена последовательность реализации максимальной защиты запросов. Во втором этапе используется установка типа для переменной, что позволит приводить данные внутри к указанному типу.

На этапе 3 происходит заготовка запроса и установка плейсхолдеров. В этом случае переменные подставляются вместо знака «?» в последний момент. Все эти пункты реализации безопасности позволили отразить некорректный запрос и отобразить страницу ошибки 404.

```
<?php
session_start();
$cruise_id = $_GET['id'];
$blacklist = Array('union', 'alter', 'update', 'delete', 'drop', 'insert', 'into', 'from', 'where',
'like','-','--','select',' ');
foreach($blacklist as $word) {
      if (stripos($cruise_id,$word) !== false)  flag = true;     }
if ( $flag == true)
{     exit(http_response_code(404));   } ?>
```

Листинг 1 – Реализация черного списка слов

```
settype($id_cruise, 'integer'); //2
$id_cruise = mysqli_real_escape_string($link, $id_cruise);
$query = "SELECT * FROM cruise where cruise_id= ?";//3
$stmt = mysqli_prepare($link, $sql);
mysqli_stmt_bind_param($stmt, 's', $id_cruise);
$result = mysqli_stmt_get_result($stmt);
```

Листинг 2 – Реализация разных уровней защиты

Таким образом, выполнена простейшая защита передачи переменных в открытом виде при помощи реализации черного списка и подготовки запроса.

ЛИТЕРАТУРА

1. Урбанович, П. П. Киберпространство: тренды, угрозы и безопасность / П. П. Урбанович // Интеграция и развитие научно-технического и образовательного сотрудничества – взгляд в будущее: сборник статей II Междунар. научно-техн. конф. "Минские научные чтения - 2019", Минск, 11-12 декабря 2019 г.: в 3 т. Т. 3. – Минск: БГТУ, 2020. – С. 180-185.

2. Урбанович, П. П. Защита информации: конспект-лекция, ч. 2 = InformationProtection, Part 2: BASICMETHODS / П. П. Урбанович. – Минск: БГТУ, 2019. – 34 с.

3. Защита от SQL инъекции, [Электронный ресурс], URL: https://htmlacademy.ru/tutorial/php/sql-injections, доступ: 10.04.2021.

4. http_response_code, [Электронный ресурс], URL: https://www.php.net/manual/ru/function.http-response-code.php, доступ: 10.04.2021.