Лабораторная работа №6

Оценка безопасности web-страниц и приложений с использованием ручного и автоматизированного анализа наличия уязвимостей типа "SQL Injection"

*Цель работы:* Освоение и систематизация знаний об уязвимостях и инструментах их выявления.

**Теоретическая часть**

Уязвимости типа "SQL Injection" ("инъекция в SQL-запросы") позволяют нарушителю выполнять несанкционированные операции над содержимым баз данных SQL -серверов путём вставки дополнительных команд в SQL -запросы. Любой SQL -запрос представляет собой последовательность команд для сервера СУБД, сформированную на основе специализированного языка структурированных запросов SQL (Structured Query Language). Данная уязвимость характерна для приложений, которые получают в качестве входных данных параметры доступа к базе данных, после чего на их основе формируют SQL -запрос к серверам СУБД. Уязвимость " SQL Injection " заключается в отсутствии проверки корректности данных, поступающих на вход программе, что потенциально может позволить нарушителю составить входные данные таким образом, что приведёт к искажению искомого SQL -запроса к СУБД.

**Поиск страниц для проведения экспериментов**

* Задать в *Google* поиск страниц с использованием служебных команд:

inurl: login.php

* Найти страницы, которые запрашивают у пользователя данные (страница поиска, обсуждений, и т.д.) и используют метод *POST*, чтобы послать команды другой Web странице; все параметры между <*FORM*> и </*FORM*> потенциально могут быть уязвимы к введению SQL кода:

<FORM action=Search/search.asp method=post>

<input type=hidden name=A value=C>

</FORM>

* Найти страницы, которые используют параметры, подобно:

http:// somesite1.ru/?ID=31610

**Проверка наличия на странице уязвимости**

* Опробовать приведенные ниже варианты в качестве входных данных в текстовые поля для ввода имени пользователя и пароля *URL*-параметра *скрытых полей*:
* hi' or 1=1--
* ' or 1=1-
* " or 1=1--
* or 1=1-
* ' or 'a'='a
* " or "a"="a
* ') or ('a'='a
* Примеры
* Login: hi' or 1=1--
* Password: hi' or 1=1
* http://somesite2/index.asp?id=hi' or 1=1--
* <FORM action=http:// somesite3/Search/search.asp method=post>
* <input type=hidden name=A value="hi' or 1=1-- ">
* </FORM>

Признаком обнаружения *уязвимости* является получение возможности неавторизованного входа на *сайт.*

**Автоматизированная проверка наличия на странице уязвимости**

* Загрузить утилиту sqlmap с сайта <http://sqlmap.org/> (там же можно найти инструкцию для работы с sqlmap) или воспользоваться сайтом <http://suip.biz/>
* Запустить утилиту, в качестве параметров передав адреса страниц, выбранных для проведения экспериментов
* Проанализировать результаты работы программы

**Действия в случае обнаружения уязвимости на странице**

При обнаружении на странице *уязвимости* *SQL Injection*

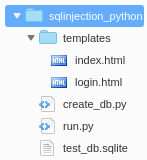
* Немедленно сообщить о найденной *уязвимости* администрации сайта
* НЕ развивать *атаку*
* НЕ размещать информацию о найденной *уязвимости* на общедоступных ресурсах
* НЕ предпринимать действий, которые могут привести к нарушению *конфиденциальности*, целостности или доступности размещенной на сайте информации

**Пример собственного web-сайта на языке Python**

1. Устанавливаем python с официального сайта (<https://www.python.org/>), желательно 3й версии.
2. Устанавливаем flask:

**$ pip install Flask**

1. Создаем файлы со следующей структурой:

**create\_db.py**

import sqlite3

# Подключаемся к базе test\_db.sqlite, в нашем случае она отсутствует. Создадим её.

conn = sqlite3.connect('test\_db.sqlite')

cursor = conn.cursor()

# Создаем таблицу пользователей

cursor.execute('CREATE TABLE users(ID INTEGER PRIMARY KEY ASC, login TEXT UNIQUE, pass TEXT)')

# Добавляем пользователя

cursor.execute('INSERT INTO users (login, pass) VALUES ("login", "pass")')

conn.commit()

conn.close()

**run.py**

import sqlite3

from flask import Flask, redirect, render\_template, session, url\_for, request

app = Flask(\_\_name\_\_)

# Секретный ключ, необходимый для сессии

app.secret\_key = 'A0Zr98j/3yX R~XHH!jmN]LWX/,?RT'

# Этот код выполняется если URL-путь пустой

@app.route('/')

def index():

# Если пользователь есть в сессии загружаем шаблон index.html

if 'username' in session:

return render\_template('index.html')

# Перенаправление на страницу login

return redirect(url\_for('login'))

@app.route('/login')

def login():

if 'username' in session:

return redirect(url\_for('index'))

return render\_template('login.html')

@app.route('/login/authentication', methods=['POST', 'GET'])

def authentication():

conn = sqlite3.connect('test\_db.sqlite')

cursor = conn.cursor()

# Пытаемся получить число совпадений с пользователем и паролем

cursor.execute("SELECT COUNT(\*) FROM users WHERE login = '%s' AND pass = '%s'" % (request.form['login'], request.form['pass'],))

res = cursor.fetchone()

conn.close()

# Если есть совпадения то добавляем имя в сессию

if res[0] != 0:

session['username'] = request.form['login']

return redirect(url\_for('index'))

return redirect(url\_for('login'))

@app.route('/logout')

def logout():

# Удаляем сессию

session.pop('username', None)

return redirect(url\_for('login'))

if \_\_name\_\_ == '\_\_main\_\_':

# host, port, debug\_mode

app.run('127.0.0.1', 80, True)

**index.html**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Главная</title>

</head>

<body>

<h1>Ура, тебе удалось войти!</h1>

{# jinja2 так позволит получить нужный URL #}

<a href="{{ url\_for('logout') }}">выйти</a>

</body>

</html>

**login.html**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Форма входа HTML5 CSS3</title>

</head>

<body>

<div class="container">

<section id="content">

{# В action записывается адрес, куда нас перенаправит после подтверждения формы. #}

<form action="/login/authentication" method="post">

<h1>Вход на сайт</h1>

<div>

<input placeholder="Имя" required="" type="text" name="login">

</div>

<div>

<input placeholder="Пароль" required="" type="password" name="pass">

</div>

<div>

<input value="Войти" type="submit">

</div>

</form>

</section>

</div>

</body>

</html>

1. Далее необходимо инициализировать базу данных:

**$ python create\_db.py**

1. После чего запускаем наш сайт командой:

**$ python run.py**

1. Заходим в браузер и вводим выбранный нами IP. Тестируем на sql-инъекции. При успешной проверке или при вводе правильных данных мы увидим следующую надпись

**Задачи:**

Выполнить одно из двух следующих заданий:

Задание 1:

* Найти сайт уязвимый для SQL-инъекций, используя соответствующие методы.
* Для этого же сайта использовать специализированную программу sqlmap для проверки на уязвимость.

Задание 2:

* Написать собственный web-сайт или приложение, которое будет уязвимо для SQL-инъекций.
* Проверить его уязвимость.
* Исправить уязвимость и убедиться, что уязвимости больше нет.