Отчет по лабораторной работе №5
Проект OWASP WebGoat

Анастасия Тарасова

8 июня 2015 г.

1 Цель работы

Изучить описание деятельности самых распространенных веб-уязвимостей согласно рейтингу OWASP.

2 Ход работы

Исследование 10 самых распространенных web-уязвимостей по рейтингу OWASP

1. **Injection** Атака на интепритатор машины-цели, позволяя выполнять произвольный код от ее имени. Чаще всего встрачаются в SQL, LDAP, Xpath, или NoSQL запросах, парсерах xml, аргументах программ и т.д.

2. **Broken Authentication and Session Management** Атака на уязвимости систем авторизации и управления сессиями с целью кражи и/или выполнения каких либо действий от чужого имени.

3. **Cross-Site Scripting** Атака на браузер путем подмены загружаемых скриптов. В результате злоумышлиниками может быть получена почти любая информация.

4. **Insecure Direct Object References** Суть атаки - изменение некого объекта, используемого в авторизированной сессии. Пример:

   ```
   String query = "SELECT * FROM accts WHERE account = ?";
   PreparedStatement pstmt = connection.prepareStatement(query , ... );

   pstmt.setString( 1, request.getParameter("acct")); <<<<<

   ResultSet results = pstmt.executeQuery( );
   ```

   Изменение параметра позволит отправлять измененные запросы от имени авторизированного пользователя.

5. **Security Misconfiguration** Ошибки в конфигурации. Атакующий может получить доступ к файлам, акаунтам, системе и т.д.

6. **Sensitive Data Exposure** Кража ценной/личной информации. Атака сложна если используется шифрование. В таком случае данные крадутся косвенными методами: на стороне клиента, когда данные уже рашифрованы, man-in-the-middle атака и другими способами.

7. **Missing Function Level Access Control** Доступ неавторизированного пользователя к привелегированным функциям. Пример:

   ```
   http://example.com/app/getappInfo
   http://example.com/app/admin_getappInfo <<<<
   ```

   Доступ к функции admin_getappInfo должен иметь только администратор. Соответственно, если пользователь, не являющийся администритором получает доступ к данной функции - это уязвимость.

8. **Cross-Site Request Forgery** Атака путем выполнения запросов к некоторому защищенному ресурсу от его имени авторизованного пользователя. Недостаток - атакующий **НЕ** может перехватить ответ от ресурса. В этом случае вводят так называемые CSRF-токены: каждый последующий пакет от клиента содержит токен, полученный в пердыдущем ответе сервера.

9. **Using Components with Known Vulnerabilities** Атака на уязвимый компонет системы, выявленный в результате сканирования.

10. **Unvalidated Redirects and Forwards** Скрытые ссылки в картинках, фреймах и т.д., ведущих на доверенный сайт. Позволяет произвести любой запрос. Пример:

    ```
    http://www.example.com/redirect.jsp?url=evil.com
    ```

Запустим уязвимое приложение WebGoat (рисунок 1).

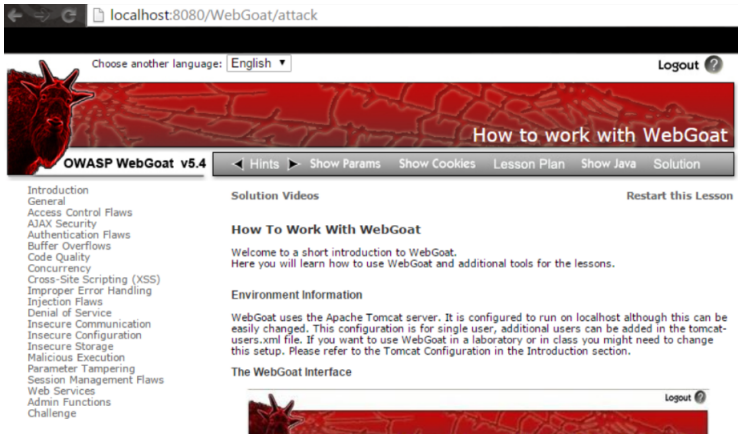

Рис. 1: Запуск WebGoat в браузере Mantra

Настроим инструмент Mantra для использования ZAP (сканера безопасновати) в качестве прокси-сервера (рисунок 2).

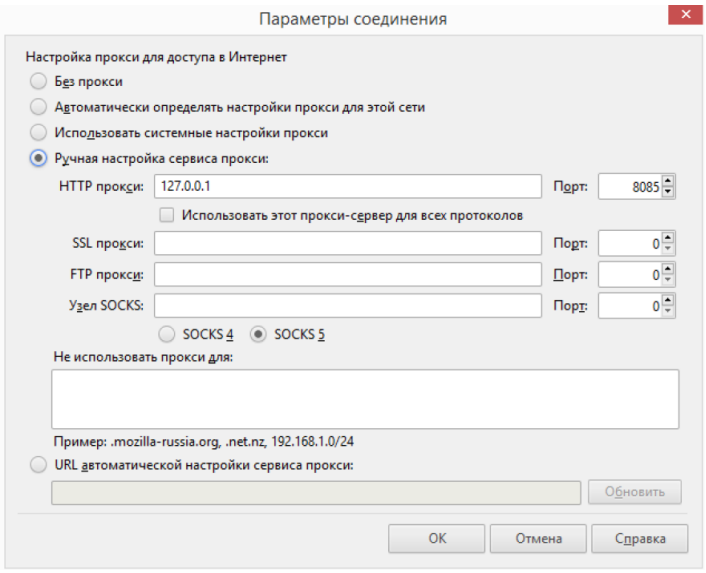

Рис. 2: Настройка прокси-сервера

Запустим ZAP и видим, что на панели сайтов появился WebGoat и перехват запросов(рисунок3).

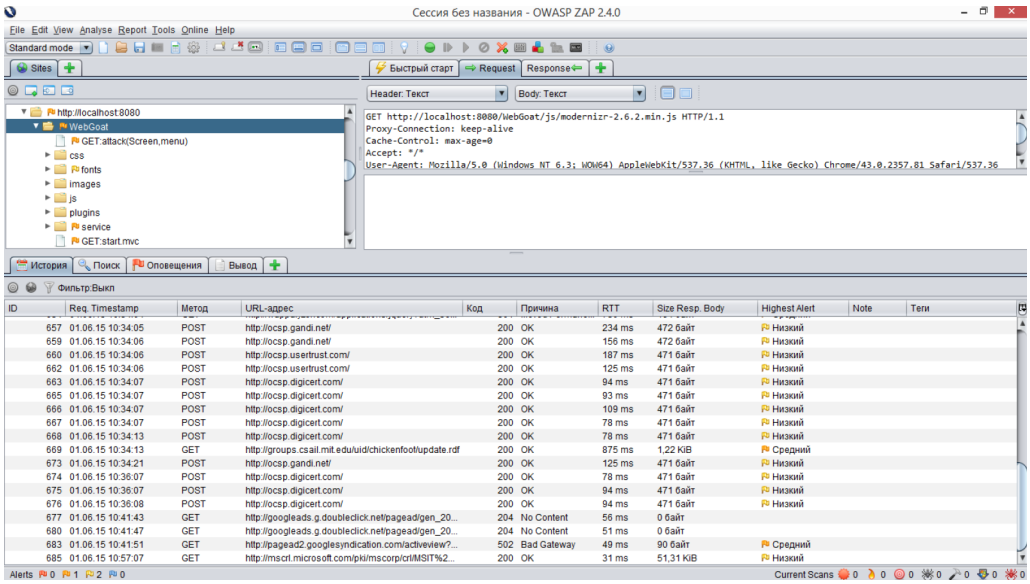

Рис. 3: Работа ZAP

Зададим Http Basics, введем своем имя в поле и поставим ZAP в режим прослушивания (рисунок 4).

Рис. 4: ZAP в режиме прослушивания

Отправим данные (GO!) и увидим, что был перехвачем POST запрос (рисунок 5.)

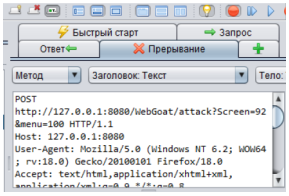

Рис. 5: ZAP перехватил POST запрос