Проект OWASP WebGoat

Абрамов Антон

5 июня 2015 г.

Содержание

1 Цель работы

2 Ход работы

Изучить описание деятельности самых распространённых веб-уязвимостей согласно рейтингу OWASP. **Практическое задание** Запустить уязвимое приложение WebGoat. Запустить сканер безопасности ZAP. Запустить инструмент Mantra, настроить его для использования ZAP в качестве прокси-сервера.

Запустим уязвимое приложение WebGoat в браузере Mantra (Рис. 1).

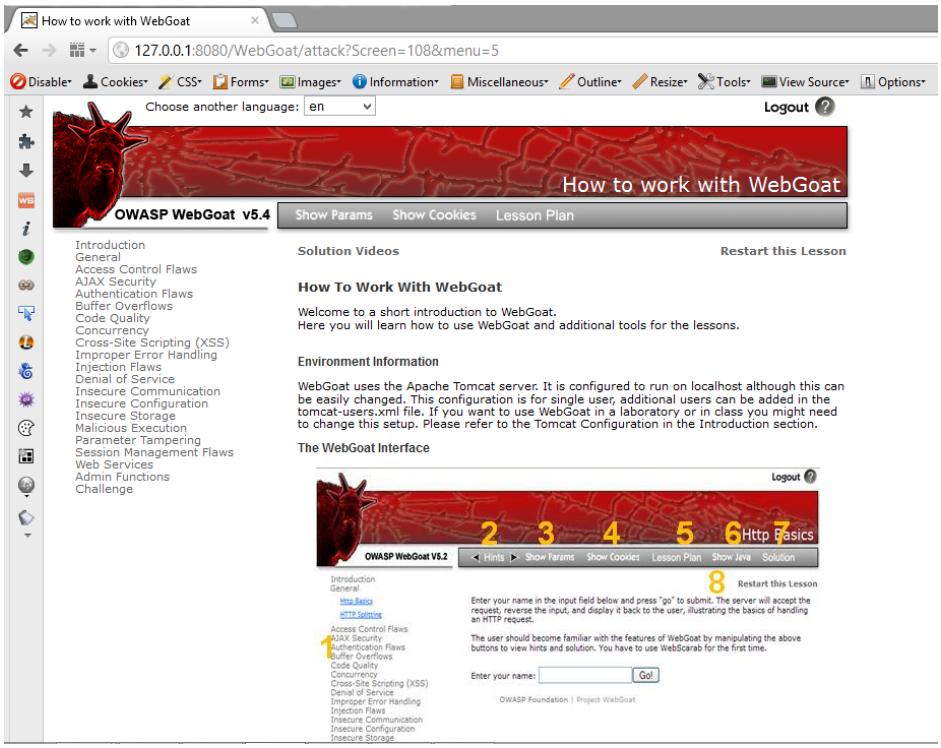

Рис. 1: WebGoat в браузере Mantra

Настроим Mantra для использования ZAP (Рис. 2).

Запустим ZAP и проверим, что все настроено верно.

Видим, что на панели сайтов появился WebGoat, а также идет перехват запросов.

Пройдемся по пунктам WebGoat.

Первый пункт General. Задание Http Basics. Не допустим переворачивания имени, вводимого в поле. Введем Anton. Рисунок 4. И перед

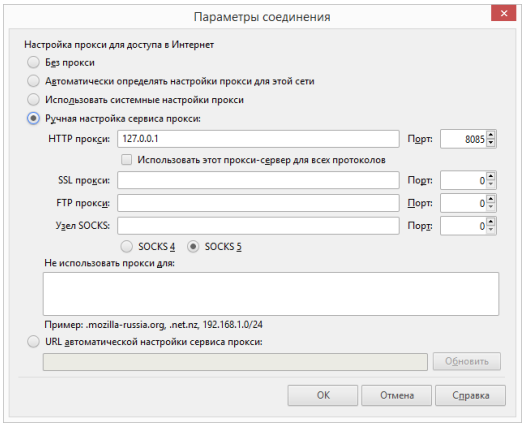

Рис. 2: Настройка прокси-сервера

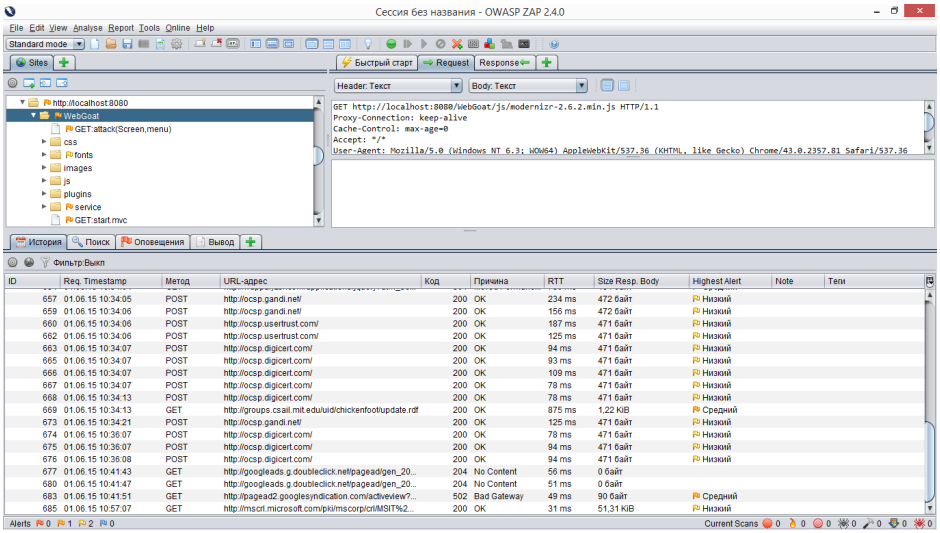

Рис. 3: Работа ZAP

отправкой данных поставим ZAP в режим прослушивания, как на рисунке 5.

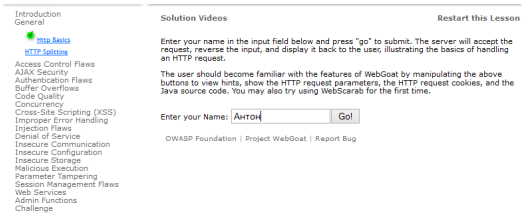

Рис. 4: Http Basics ввод данных

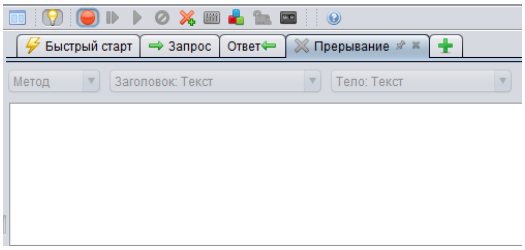

Рис. 5: ZAP в режиме прослушивания

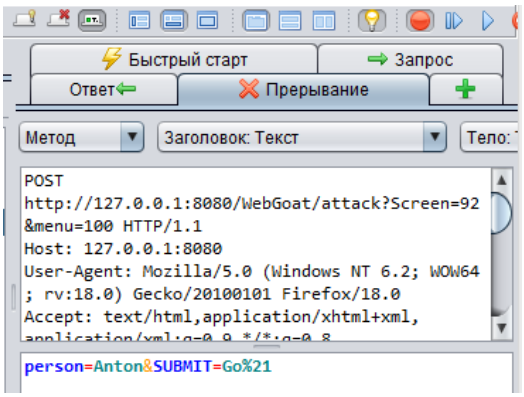

Рис. 6: Перехваченный POST запрос в ZAP

Отправим данные нажатием Go! и посмотрим, что поймал ZAP. На рисунке 6 видно, что был перехвачен POST запрос. Подменим введенные данные на notnA, когда строка будет перевернута получим ту же строку, что и отправляли. Урок пройдет. Далее не будет подробно описываться, когда ZAP выставляется на прослушивание. Если присутствует скриншот с перехваченными данными ZAP, значит был выставлен режим прослушивания. Также не будут рассмотрены все уроки, так как их большое количество. Из каждого урока будет рассмотрен один интересный пример, заслуживающий большего внимания среди основных.

Поехали!!!

1. Недостатки контроля доступа

   Сначала мы просматриваем, какие права доступа выданы пользователям, проверяем, все ли настройки соответствуют реальности, в итоге оказывается, что Larry имеет слишком большие возможности, чего быть не должно. Далее удалим администратора из-под учетной записи пользователя, у которого нет прав на удаление. Зайдем под Томом. Выберем пункт просмотра данных и подменим функцию просмотра на функцию удаления, как на рисунке 7.

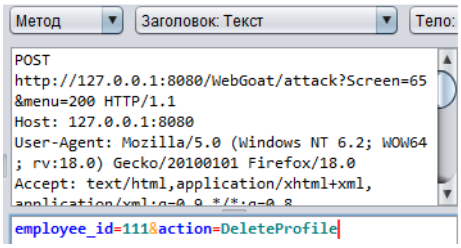

Рис. 7: Подмена функции просмотра на удаление

Таким образом вместо просмотра собственного пользователя вы удалили аккаунт пользователя с id = 111.

Таким же способом можно просмотреть профиль любого пользователя.

2. Безопасность AJAX

ajax запросы выполняются на лету, без перезагрузки страницы. Это очень удобно, но есть множество уязвимостей, о которых надо знать.

Например, мы вводим вместо своего имени сторонний html код и вместо приветствия выводит картинку, как на рисунке 8.

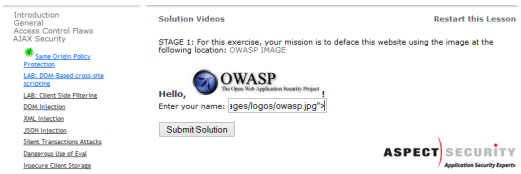

Рис. 8: Ввод стороннего html кода.

Необходимо следить за тем, чтобы ajax не выгружал лишнюю информацию в код, иначе ее сможет просмотреть любой пользователь, что не есть хорошо.

При этом необходимо делать валидацию и проверку данных не только на клиенте, но и на сервере, что не дает злоумышленникам возможность подменять ajax запросы.

Также можно подменить ответ на ajax-запрос, это тоже надо учитывать. Вернем больше информации, чем требовалось.

В обычном случае наш выбор ограничен, что и видно из рисунка 9.

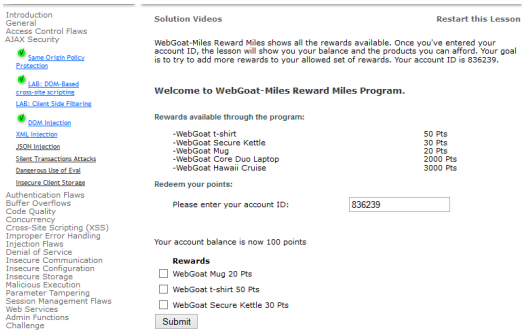

Рис. 9: Ограниченный выбор

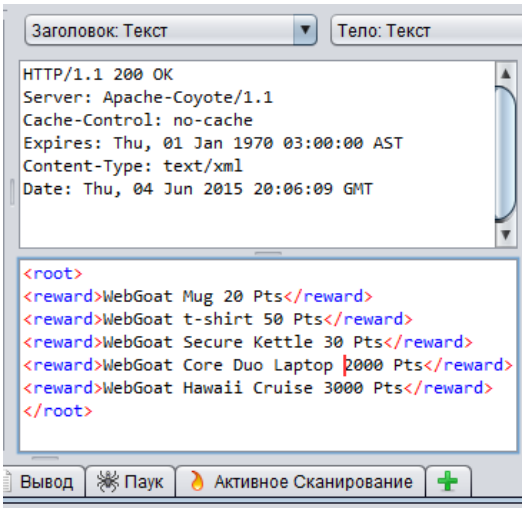

Рис. 10: Подмена ответа на ajax-запрос

Перехватим ответ на ajax-запрос и подменим результат (Рис. 10).

Как видно из рисунка 11, у нас появились новые пункты выбора, которых не было до этого.

3. Недостаток аутентификации

   Первое о чем стоит сказать, это необходимость выбирать сложные пароли для защиты своих аккаунтов. Использовать в своем пароле не только цифры, но и буквы с разным регистром, что значительно усложняет пароль.

   Стоит обратить особое внимание секретные вопросы, для восстановления пароля. Не выбирайте слишком легки вопросы и очевидные ответы на них.

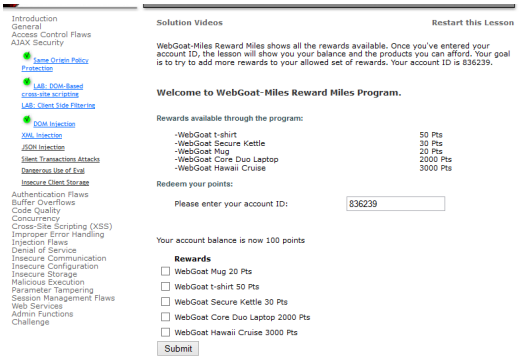

Рис. 11: расширенная возможность выбора

Данные об аутентификации могут храниться в заголовках http сеанса, чем может воспользоваться злоумышленник.

4. Переполнение буфера

   Всегда необходимо следить за размерностью переменной, в которую записывается введенное значение. Из-за переполнения буфера может сломаться логика защиты, и злоумышленник получит доступ к закрытой информации.

5. Качество кода

   Всегда необходимо следить за тем, что остается в комментариях к коду, который может увидеть пользователь. Не стоит туда записывать какую-либо системную или конфиденциальную информацию. В крайнем случае необходимо следить за тем, чтобы эта информация была удалена в версии релиза и не попала на боевой сервер.

   Один из таких примеров отображен на рисунке 12.

6. Многопоточность

   Всегда необходимо учитывать, что всегда может осуществляться несколько попыток авторизации. При этом надо проверять, какой пользователь авторизован в данный момент, причем эту проверку необходимо делать не на клиенте, а на сервере.

   На рисунке 13 видно, что в прайсе стоит одна сумма, а в итоге получилась совершенно другая, а все дело в неправильной реализации логики покупки и неверной проверке авторизированного пользователя.

7. Межсайтовое выполнение сценариев

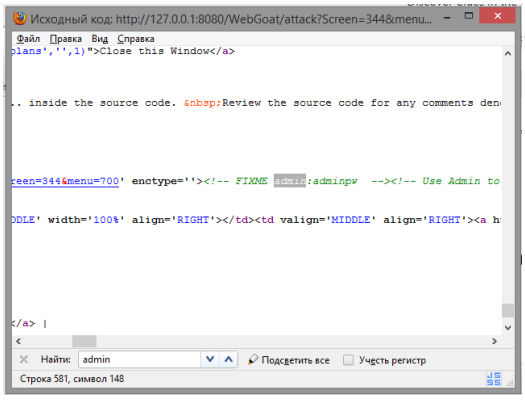

Рис. 12: Конфиденциальная информация в коде

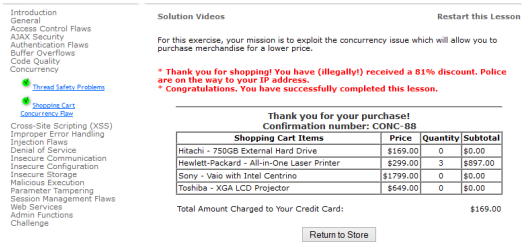

Рис. 13: Ошибка многопоточности в цене

XSS и HTML вставки можно использовать для подделки настоящего блока на фиктивный, выполняющий сбор данных о пользователе. При этом пользователь даже не подозревает, что страница не настоящая и доверяет ей.

Из рисунка 14 видно, что изначально была просьба ввести имя, добавив в это поле совершенно иное содержимое появилась форма ввода логина и пароля.

Подобный код можно вставлять в любые поля ввода, что может испортить работу приложения. Чтобы защитить себя от подобных вещей, необходимо преобразовывать введенные данные.

<%=lesson.htmlEncode(webSession, employee.getAddress1())%>

8. Неправильная обработка ошибок

   Необходимо правильно обрабатывать ошибки при аутентификации пользователей, чтобы нельзя было зайти под пользователем, полностью игнорируя ввод пароля, как это сделано на рисунке 15.

   Перехватим такой запрос и заменим **Username=webgoat&Password**

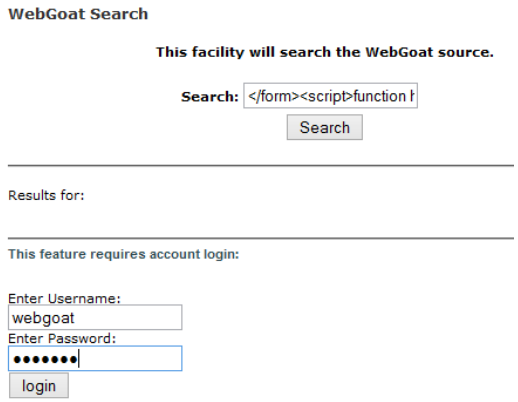

Рис. 14: Использование XSS

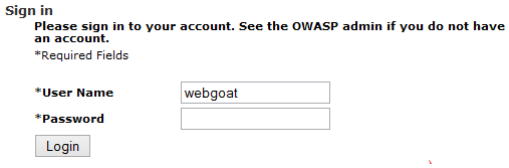

Рис. 15: Игнорирование ввода пароля

=&SUBMIT=Login на **Username=webgoat&SUBMIT=Login**. Если обработка ошибок при аутентификации неверна, то можно получить доступ к данной учетной записи.

9. Недостатки, приводящие к осуществлению инъекций (SQL и прочее)

   SQL инъекции одна из самых распространенных web-атак. SQL инъекции представляют собой серьезную угрозу для любой базы данных.

   Можно вместо данных ввести такую строку, которая будет нарушать производимый запрос и делать совершенно иные действия.

10. Отказ в обслуживании

    Можно попытаться задосить сервер. Одним из таких примеров - это ограниченный пул соединений с БД. В данном случае, если ввести количество логинов больше чем 2, то приложение может сломаться (Рис. 16).

11. Небезопасность сетевого взаимодействия

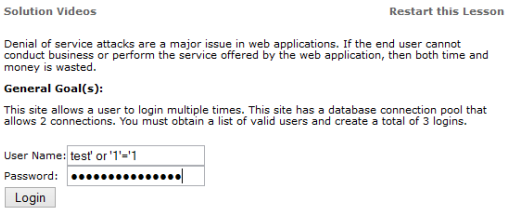

Рис. 16: Отказ в обслуживании

Даже если пароль скрывается при вводе, его можно перехватить. Это возможно, когда пароль передается по незащищенному или плохо защищенному каналу связи.

Перехватив запрос аутентификации, мы можем получить пароль пользователя и использовать его в своих целях.

Для использования защищенного канала необходимо использовать https соединение и использовать протокол TLS.

12. Небезопасная конфигурация

    У многих сайтов есть страница конфигурации или админка. Рекомендуется делать неочевидным адрес таких страниц, это позволит избежать попыток взлома. В примере, изображенном на рисунке 17 страница конфигурации расположена по достаточно простому о очевидному адресу.

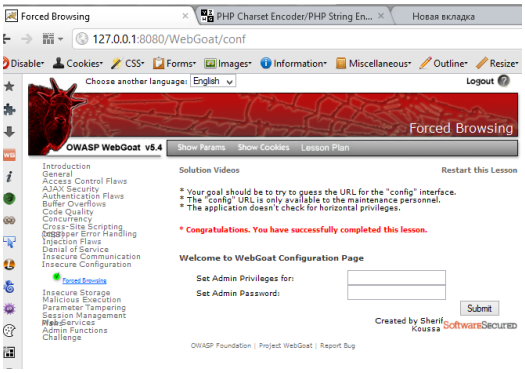

Рис. 17: Адрес страницы конфигурации

13. Небезопасное хранилище

    Необходимо заботиться не только о безопасной передаче данных, но и о безопасном хранении этих данных на серверах. Все больше компании в последнее время задумывается об этом. В данном уроке

мы посмотрели, как выглядит наш пароль в разных кодировках и в зашифрованном виде (Рис. 18).

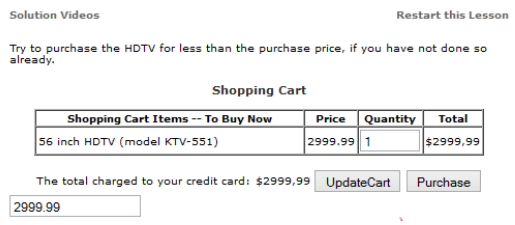

Рис. 18: Информация в разном виде

14. Использование злонамеренного кода

    Зная в какой директории хранятся файла на сервере, можно загрузить файл и запустить его на выполнение.

15. Подделка параметров

    Совершим покупку (Рис. 19) дешевле, чем она стоит на самом деле.

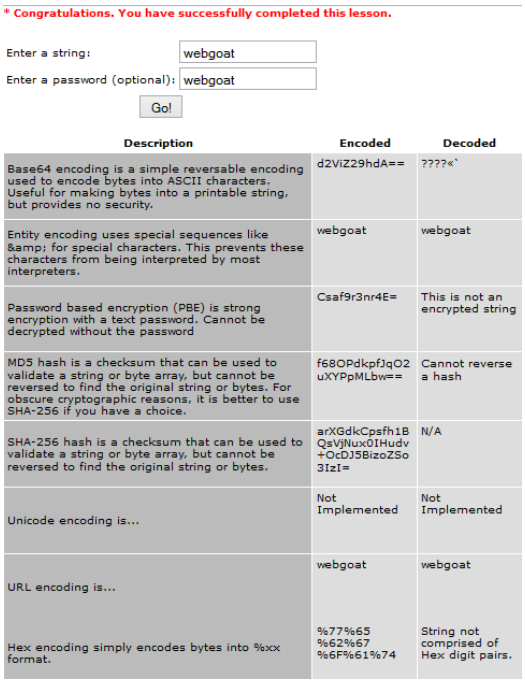

Рис. 19: Совершение покупки

Перехватим данные и заменим **QTY=1&SUBMIT=Purchase&Price=2999.99** на **QTY=1&SUBMIT=Purchase&Price=9.99**

Результат выполнения покупки приведен на рисунке 20.

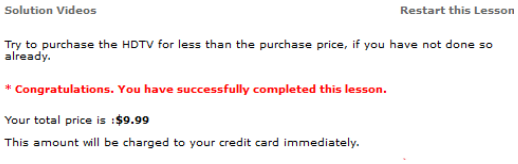

Рис. 20: Результат покупки

Другой пример, это ухитриться обойти проверку введенных данных на клиенте и отправить неверные данные на сервер. Результат такого действия представлен на рисунке 21.

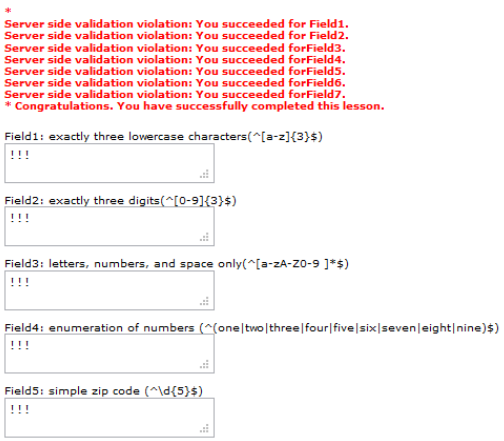

Рис. 21: Отправка на сервер неверных данных

16. Недостатки управления сессией

    Необходимо быть очень аккуратным при разработке собственных идентификаторов сессии. Можно упустить важные детали, тем самым дать возможность злоумышленникам воспользоваться этим.

    Можно отправить электронное письмо жертве, которое выглядит как официальное. Добавим в это письмо шаблонное сообщение, содержащее идентификатор сессии, как это сделано на рисунке 22.

17. Безопасность веб-сервисов

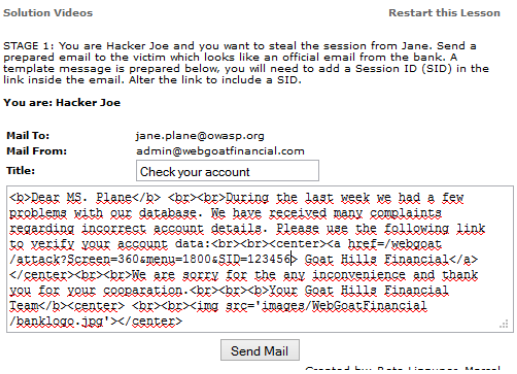

Рис. 22: Письмо с идентификатором сессии

3 Выводы

В ходе выполнения данной лабораторной работы были рассмотрены распространённые веб-уязвимости и методы по из устранению.