

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Кафедра Компьютерных Систем и Программных Технологий

ОТЧЕТ

по лабораторной работе №8

Тема: «Проект OWASP WebGoat»

Дисциплина: «Методы и средства защиты информации»

Выполнил: студент гр. 53501/2
Пономарев М.А.

Преподаватель
Вылегжанина К.Д.

Санкт-Петербург
2015

Содержание

1	Задание	2
2	Выполнение работы	3
2.1	Изучение	3
2.1.1	Описания десяти самых распространенных веб-уязвимости согласно рейтингу OWASP	3
2.2	Запуск WebGoat, ZAP, Mantra	5
2.3	Basics	6
2.4	Недостатки контроля доступа	7
2.5	Безопасность AJAX	8
2.6	Недостатки аутентификации	10
2.7	Переполнение буфера	10
2.8	Качество кода	10
2.9	Многопоточность	10
2.10	Межсайтовое выполнение сценариев	11
2.11	Неправильная обработка ошибок	11
2.12	Недостатки приводящие к осуществлению инъекций	11
2.13	Отказ в обслуживании	12
2.14	Небезопасное сетевое взаимодействие	12
2.15	Небезопасная конфигурация	13
2.16	Небезопасное хранилище	13
2.17	Исполнение злонамеренного кода	13
2.18	Подделка параметров	15
2.19	Недостатки управление сессией	15
2.20	Безопасность веб-сервисов	15
3	Выводы	17

1 Задание

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

- а) Недостатки контроля доступа
- б) Безопасность AJAX
- в) Недостатки аутентификации
- г) Переполнение буфера
- д) Качество кода
- е) Многопоточность
- ж) Межсайтовое выполнение сценариев
- з) Неправильная обработка ошибок
- и) Недостатки приводящие к осуществлению инъекций (SQL и прочее)
- к) Отказ в обслуживании
- л) небезопасное сетевое взаимодействие
- м) небезопасная конфигурация
- н) небезопасное хранилище
- о) Исполнение злонамеренного кода
- п) Подделка параметров
- р) Недостатки управления сессией
- с) Безопасность веб-сервисов

2 Выполнение работы

2.1 Изучение

2.1.1 Описания десяти самых распространенных веб-уязвимости согласно рейтингу OWASP

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

— **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.

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

— **Insecure Direct Object References** Суть атаки - изменение некоего объекта, используемого в авторизированной сессии.

Изменение параметра позволит отправлять измененные запросы от имени авторизованного пользователя.

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

— **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 расшифрованы, man-in-the-middle атака и другими способами.

— **Missing Function Level Access Control** Доступ неавторизованного пользователя к привелегированным функциям. Пример:

```
\begin{Verbatim}[frame=single]
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 администратором получает доступ к данной функции - это уязвимость.

— **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 предыдущем ответе сервера.

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

— **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`

2.2 Запуск WebGoat, ZAP, Mantra

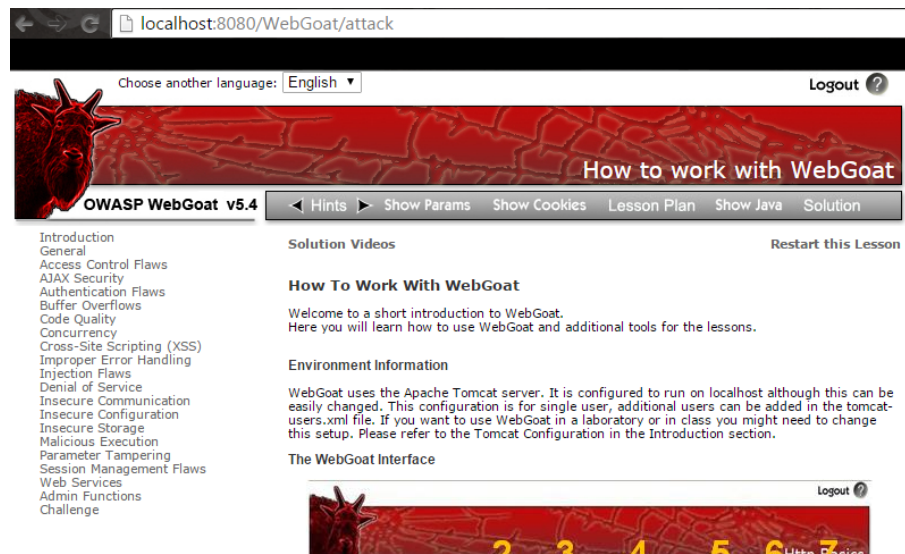


Рисунок 1 — WebGoat запущен на порту 8080. Системные настройки http прокси-сервера установлены в localhost:8081

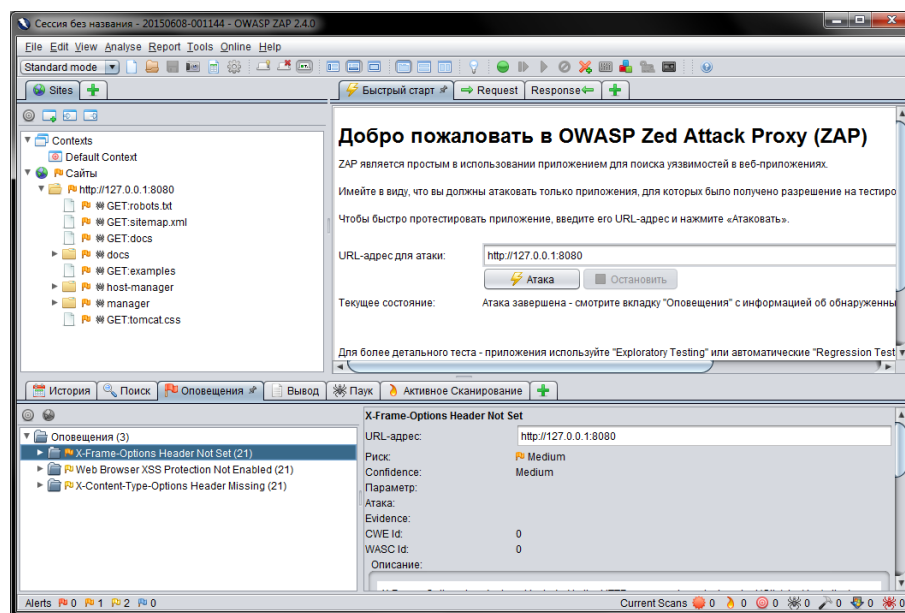


Рисунок 2 — ZAP запущен. Поднят локальный прокси-сервер на порту 8081.

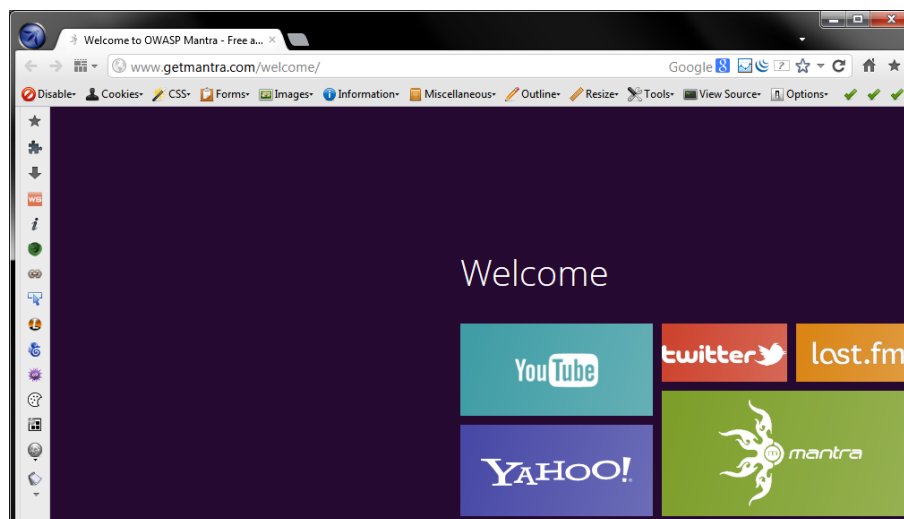


Рисунок 3 — Mantra запущен. Переход на любой сайт отображается в ZAP в виде перехваченных данных. Все работает нормально.

2.3 Basics

Сначала переводим ZAP в режим прослушивания и перехвата, нажав плюсик рядом со вкладкой Response, а затем нажав кнопку Set break on all requests.

```
POST http://127.0.0.1:8080/WebGoat/attack
Host: 127.0.0.1:8080
Proxy-Connection: keep-alive
Content-Length: 24
Cache-Control: max-age=0
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Accept: text/html,application/xhtml+xml,a
Origin: http://127.0.0.1:8080
person=Igor&SUBMIT=Go%21
```

Рисунок 4 — В ZAP видим перехваченные данные. Чтобы имя не было перевернуто, изменим его в ZAP на rogl и отправим. Урок пройден.

2.4 Недостатки контроля доступа

* Congratulations. You have successfully completed this lesson.
* User Larry [User, Manager] was allowed to access resource Account Manager

Change user:

Select resource:

Рисунок 5 — Находим пользователя Larry, у которого есть доступ к ресурсу Account Manager, хотя его быть не должно.

Во второй части этого урока мы получаем доступ к файлу, находящемуся вне нашей директории. Для этого перехватим запрос как это было сделано ранее. В списке выберем любой файл и нажмем View File. В окне ZAP заменим имя файла на строку `"../../../../conf/tomcat-users.xml"`. Доступ к файлу получен.

В третьей части урока заходим под аккаунтом админа John:john и выясняем, что метод для удаления пользователей называется DeleteProfile. Затем заходим под аккаунтом Tom:tom и перехватывая запрос View Profile подменим вызываемый метод на DeleteProfile. Профиль другого сотрудника удален. Второй и четвертый шаги не выполнить, поскольку версия WebGoat не Developers. На третьем шаге авторизируемся под аккаунтом Tom:tom и выбирая в списке себя, нажимаем кнопку View Profile. Перехватывая этот запрос в ZAP подменим аргумент `employee_id` на другой, например, 107. Информация о другом сотруднике получена.

В четвертой части урока нам надо попытаться получить информацию, доступную только админу. Заходя на вкладки User Information и Product Information мы добавляем в адресную строку к нашему запросу еще один аргумент: `admin=true`.

* Congratulations. You have successfully completed this lesson. * Congratulations. You have successfully completed this lesson.

USERID	USER_NAME	PASSWORD	COOKIE
101	jsnow	passwd1	
102	jdoe	passwd2	
103	jplane	passwd3	
104	jeff	jeff	
105	dave	dave	

PRODUCTID	PRODUCT_NAME	PRICE
14365	56 inch HDTV	\$6999.99
24569	60 GB Hard Drive	\$149.99
32226	Dog Bone	\$1.99
35632	DVD Player	\$214.99
56970	80 GB Hard Drive	\$179.99

Рисунок 6 — Доступ к User Information получен.

2.5 Безопасность AJAX

Первая часть урока объясняет почему запросы могут отправляться только серверу-отправителю страницы. Это сделано для повышения безопасности, но может быть отключено. Современные браузеры блокируют загрузку сторонних скриптов.

Вторая часть урока наглядно показывает, что необходимо экранировать поля ввода. В данном случае можно легко подменить текст произвольным html кодом.

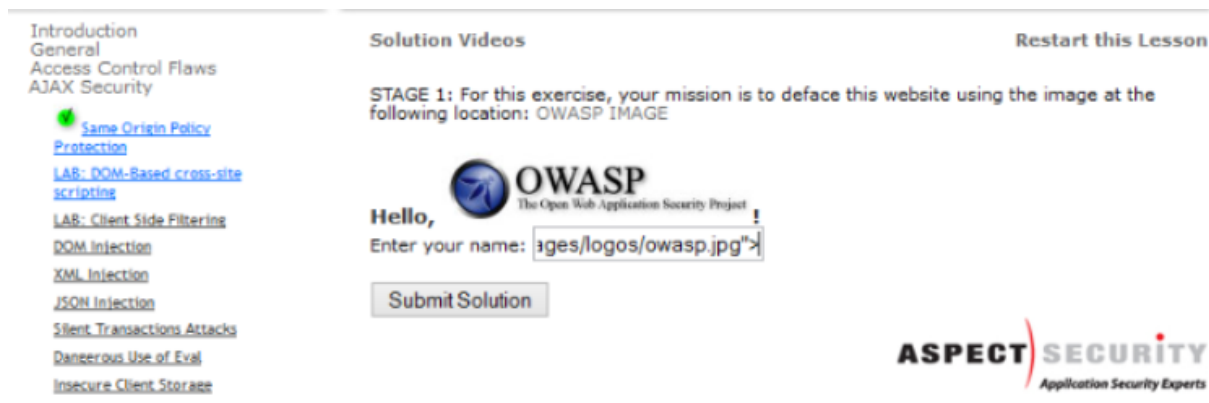


Рисунок 7 — Заменили имя на ссылку на картинку.

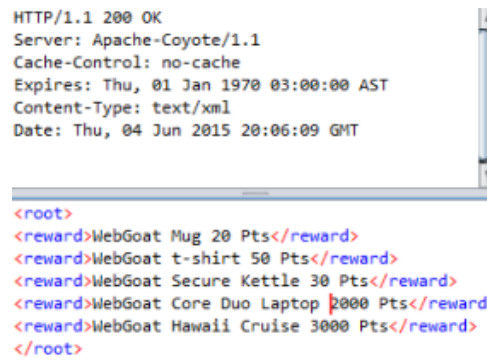
Третья часть урока показывает, что необходимо делать валидацию данных не только на клиенте, но и на сервере, чтобы исключить подмену ajax-запросов. Изменив скрипт обработки запроса мы исключили возможность выдачи лишних данных клиенту.

Четвертая часть урока поясняет принцип работы DOM injection. Используя инъекцию меняем свойство кнопки submit.disabled на false при помощи перехвата запроса и инъекции следующей строки:

```
document.forms[0].SUBMIT.disabled=false;
```

Кнопка становится активной.

Пятая часть урока затрагивает XML injection. Нам предлагается выбрать себе награду, введя свой ID. Запускаем ZAP, перехватывает запрос, вводя свой номер. Правим XML-код, добавляя себе награды используя тег <reward></reward>.



The screenshot shows a web browser window. The top part displays the HTTP response headers: HTTP/1.1 200 OK, Server: Apache-Coyote/1.1, Cache-Control: no-cache, Expires: Thu, 01 Jan 1970 03:00:00 AST, Content-Type: text/xml, and Date: Thu, 04 Jun 2015 20:06:09 GMT. Below the headers, the XML content is displayed, showing a root element with five reward items: WebGoat Mug 20 Pts, WebGoat t-shirt 50 Pts, WebGoat Secure Kettle 30 Pts, WebGoat Core Duo Laptop 2000 Pts, and WebGoat Hawaii Cruise 3000 Pts.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 03:00:00 AST
Content-Type: text/xml
Date: Thu, 04 Jun 2015 20:06:09 GMT

<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>
```

Рисунок 8 — Добавляем себе наград.

Шестая часть урока про JSON injection. Аналогично предыдущим вариантам подменяем JSON код, перехватывая запрос. Выбираем лучшую цену.

Седьмая часть - не стоит делать проверку на клиентской стороне. Необходимо найти клиентскую функцию для отправки и вызвать ее.

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

Восьмая часть аналогична: убираем readonly и отмечаем GOLD, стоимость покупок = 0.

В девятой части описывается опасность использования функции eval. Вводим в поле код

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

2.6 Недостатки аутентификации

Абсолютно очевидно, что сложность подбора пароля зависит от его длины и набора символов. Вторая часть урока показывает, что слишком простые способы восстановления делают любой пароль уязвимым. Можно с легкостью подобрать ответ на секретный вопрос типа "Ваш любимый цвет".

Часть Basic Authentication показывает как можно легко расшифровать содержимое заголовка, раскодировав содержимое его значения из base64 в простой текст. Получили guest:guest. В настройках браузера чистим куки и данные сессий и логинимся как basic:basic.

В третьей части плохо реализованная многоуровневая защита. Перехватываем данные, изменяем hiddentan=1. Обошли защиту.

В четвертой части опять многоуровневая защита, авторизация под аккаунтом Джо, вводим его tan, Перехватываем данные, в запросе указываем Jane.

2.7 Переполнение буфера

Перехватываем пакет с помощью ZAP, добавляем в значение аргумента goomno >4096 символов.

		13131313131313131313131313131313
Johnathan	Ravem	4321
John	Smith	56
Ana	Armeta	78
Lewis	Hamilton	9901

We would like to thank you for your payment.

Рисунок 9 — Запоминаем какого-либо пользователя, заходим от его имени.

2.8 Качество кода

В коде страницы можно найти логин и пароль администратора: admin:adminpw.

2.9 Многопоточность

В первой части при одновременном получении данных пользователя возможна их утечка. Можно ополучить чужие данные. Открываем два окна и вводим имена пользователей. В некоторых ситуациях можно получить не свою информацию.

Во второй части открываем два окна, в одном делаем большую покупку, в другом- маленькую, Продолжаем маленькую покупку и обновляем большую. При подтверждении платим за маленькую, а получаем большую покупку.

2.10 Межсайтовое выполнение сценариев

С помощью XSS и HTML можно заменять элементы страницы на фиктивные, а пользователь даже не поймет что что-то изменилось.

WebGoat Search

This facility will search the WebGoat source.

Search:

Results for:

This feature requires account login:

Enter Username:

Enter Password:

Рисунок 10 — Использование XSS.

2.11 Неправильная обработка ошибок

В перехваченном пакете достаточно удалить поле пароля и авторизация пройдет успешно.

2.12 Недостатки приводящие к осуществлению инъекций

— Command injection Перехватываем запрос, добавляем к имени файла строку:

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

— Numeric SQL injection Перехватываем запрос. Модифицируем:

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

— Log spoofing Перехватываем запрос, меняем имя на следующее:

```
somename
```

```
Admin succefully entered!
```

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

— SQL Injection Перехватываем сообщение. В качестве пароля вводим:

```
123123%27%200R%20%271%27%3D%271
```

При попытке получить данные на втором шаге получаем сообщение:

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

— String sql injection Аналогично. Вместо имени вводим 123123' OR 'a' = 'a'. Получаем все возможные значения.

— Modify Data with SQL INJECTION Вместо имени вводим:

```
123'; UPDATE salaries SET salary=5  
WHERE userid='jsmith
```

— Database backdoors По такой же схеме можно добавлять и триггеры:

```
101; CREATE TRIGGER myBackDoor BEFORE INSERT ON  
employee FOR EACH ROW BEGIN UPDATE employee  
SET email='john@hackme.com' WHERE userid = NEW.userid
```

2.13 Отказ в обслуживании

В пункте ZipBomb создается файл zip, содержащий большое количество одинаковых символов. При распаковке потребление памяти станет огромным из-за большой степени сжатия.

В пункте DoS from Multiple logins вместо пароля вводим

```
rewrew' or '1' or '1
```

Получаем

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

Авторизуемся этими данными. Получаем отказ в обслуживании из-за большого количества сессий.

2.14 небезопасное сетевое взаимодействие

При передаче пароля вне защищенного соединения его можно легко перехватить, для избежания этого необходимо использовать https + TLS. Пароль можно получить перехватив запрос аутентификации.

2.15 Небезопасная конфигурация

У большинства сайтов есть панель администрирования. Если она будет расположена в очевидном месте, можно легко получить к ней доступ, следовательно необходимо скрыть ее местонахождение. В данном случае она находится в директории WebGoat/conf.

2.16 Небезопасное хранилище

Различные кодировки строк, лучше хранить данные в зашифрованном виде. Тогда даже при утечке с ними сложно что-либо сделать.

Enter a string:

Enter a password (optional):

Description	Encoded	Decoded
Base64 encoding is a simple reversible encoding used to encode bytes into ASCII characters. Useful for making bytes into a printable string, but provides no security.	MTIz	?m?
Entity encoding uses special sequences like & for special characters. This prevents these characters from being interpreted by most interpreters.	123	123
Password based encryption (PBE) is strong encryption with a text password. Cannot be decrypted without the password	ZyHX1aXEh0k=	This is not an encrypted string
MD5 hash is a checksum that can be used to validate a string or byte array, but cannot be reversed to find the original string or bytes. For obscure cryptographic reasons, it is better to use SHA-256 if you have a choice.	ICy5YqxZB1uWSw cVLSNLcA==	Cannot reverse a hash
SHA-256 hash is a checksum that can be used to validate a string or byte array, but cannot be reversed to find the original string or bytes.	pmWkWSBCL51Bf khn79xPuKBKHz// H6B+mY6G9/eieu M=	N/A

Рисунок 11 — Строка "123" в различных кодировках.

2.17 Исполнение злонамеренного кода

Если злоумышленнику известна директория с исполняемыми файлами, то он может загрузить туда свой код и исполнить его. Загрузим на сервер скрипт со следующим содержанием

```
<html>
<%
java.io.File fil = new java.io.File(''D:\WebGoat-5.4\tomcat
\webapps\WebGoat\mfe_target\guest.txt'');
file.createNewFile();
%>
</html>
```

2.18 Подделка параметров

В части Bypass HTML Field Restrictions необходимо в перехваченном сообщении поменять значение всех полей и добавить disabledinput.

Часть Exploit Hidden Fields выполняется аналогично.

Третья часть выполняется аналогично: в поле сообщения вводим тело скрипта

```
<script>alert("hey")</script>
```

Для отправки скрипта на другой адрес меняем в перехваченном пакете значение аргумента to, чтобы получился такой код

```
gId=GMail+id&gPass=password&subject=Comment+for+WebGoat&
to=friend%40owasp.org&msg=%3Cscript%3Ealert%28%22hey%22
%29%3Cscript%3E&SUBMIT=Send%21
```

Bypass Client Side JavaScript Validation выполняется таким е способом.

2.19 Недостатки управление сессией

При перехвате пакетов получаем два ключа

```
65432ubphcfx
```

```
65432udfqtb
```

Нетрудно догадаться, что ключи получаются путем прибавления числа 65432 в начало и сдвига всех букв на 1 вперед, причем имя записывается задом наперед. Получается, что для логина alice ключ будет 65432fdjmb. Далее перехватываем пакет и меняем заголовок Cookie AuthCookie=65432fdjmb.

Hijack a Session представляет собой более сложную версию предыдущего задания без использования Cookie AuthCookie, но с использованием Cookie WEAKID.

Session fixation - посылаем ложное электронное письмо, в ссылку в письме добавим любой SID.

Когда жертва "залогинится" по ссылке в письме, у нас будет активная сессия, номер которой был послан в письме.

2.20 Безопасность веб-сервисов

Сервис WSDL запустить не удалось.

You are: Hacker Joe

Mail To: jane.plane@owasp.org
Mail From: admin@webgoatfinancial.com
Title:

```
<b>Dear MS. Plane</b> <br><br>During the last week we had a few
problems with our database. We have received many complaints
regarding incorrect account details. Please use the following link
to verify your account data:<br><br><center><a
href=/webgoat/attack?Screen=132&menu=1800&SID=123> Goat Hills
Financial</a></center><br><br>We are sorry for the any
inconvenience and thank you for your cooperation.<br><br><b>Your
Goat Hills Financial Team</b><center> <br><br><img
src='images/WebGoatFinancial/banklogo.jpg'></center>
```

Рисунок 12 — Посылаем ложное письмо.

3 Выводы

Инструментарий WebGoat позволяет на практике изучить основные уязвимости в работе веб-приложений и получить знания о том, как их предотвращать и не допускать при разработке реальных проектов.