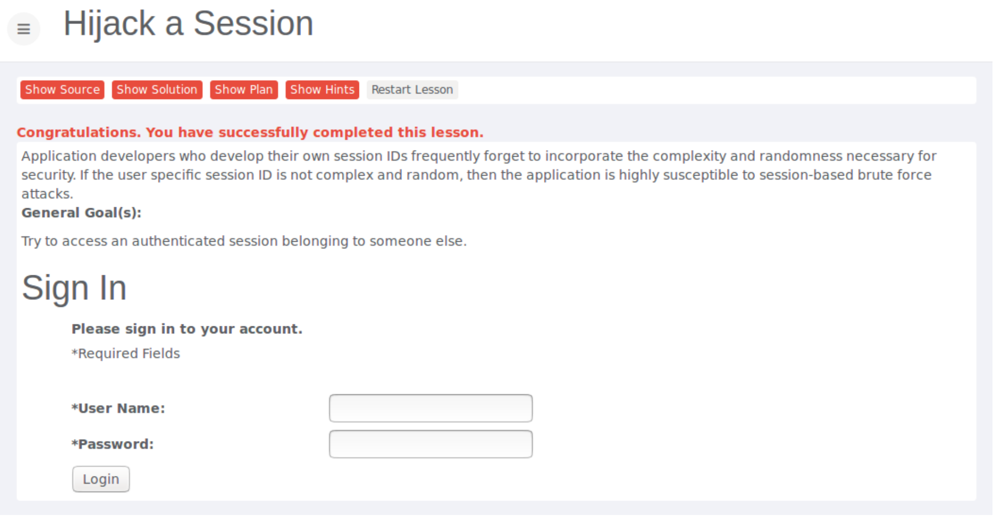
Web Security
Assignment 2

Christoph Schmidl
s4226887
Data Science
c.schmidl@student.ru.nl

April 18, 2017

1. Do the following two lessons in WebGoat (you do not have to submit any notes regarding these exercises to blackboard):

   - Session Management Flaws - Hijack a Session



Hijack a Session

Congratulations. You have successfully completed this lesson.

Application developers who develop their own session IDs frequently forget to incorporate the complexity and randomness necessary for security. If the user specific session ID is not complex and random, then the application is highly susceptible to session-based brute force attacks.

General Goal(s):

Try to access an authenticated session belonging to someone else.

Sign In

Please sign in to your account.

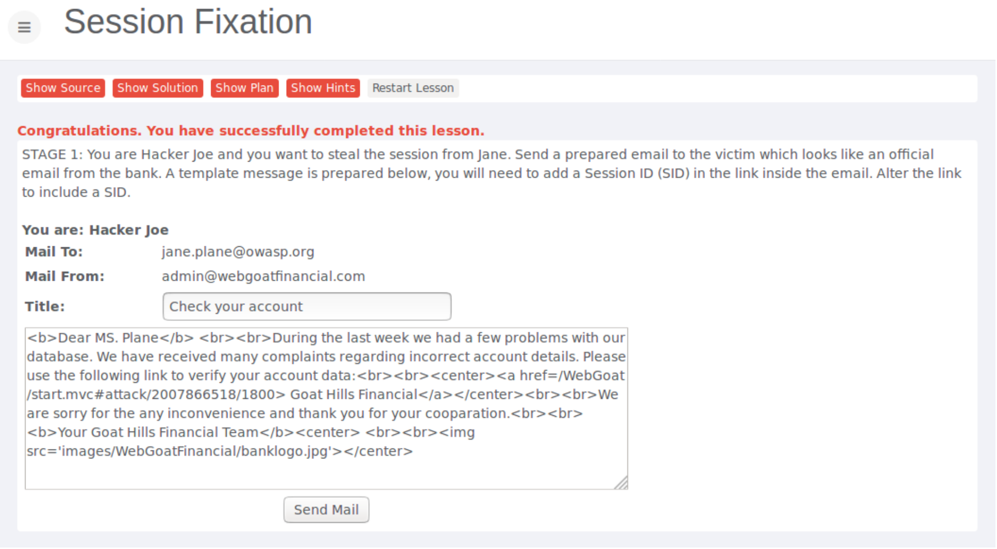
*Required Fields

***User Name:**

***Password:**

Session hijacking, sometimes also known as cookie hijacking is the exploitation of a valid computer session - sometimes also called a session key - to gain unauthorized access to information or services in a computer system. In particular, it is used to refer to the theft of a magic cookie used to authenticate a user to a remote server. In this particular lesson the value of the cookie named ”WEAKID” can be bruteforced by using a tool like JHijack. To find a value to start with you can use the SessionID Analysis Tab of WebScarab. You do not know the identity of the user's session you are going to hijack beforehand.

   - Session Management Flaws - Session Fixation



Session Fixation

Congratulations. You have successfully completed this lesson.

STAGE 1: You are Hacker Joe and you want to steal the session from Jane. Send a prepared email to the victim which looks like an official email from the bank. A template message is prepared below, you will need to add a Session ID (SID) in the link inside the email. Alter the link to include a SID.

You are: Hacker Joe

Mail To: jane.plane@owasp.org

Mail From: admin@webgoatfinancial.com

Title: Check your account

```
<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
/start.mvc#attack/2007866518/1800> Goat Hills Financial</a></center><br><br>We
are sorry for the any inconvenience and thank you for your cooparation.<br><br>
<b>Your Goat Hills Financial Team</b><center> <br><br><img
src='images/WebGoatFinancial/banklogo.jpg'></center>
```

Session fixation attacks attempt to exploit the vulnerability of a system that allows one person to fixate (find or set) another person's session identifier. Most session fixation attacks are web based, and most rely on session identifiers being accepted from URLs (query string) or POST data. A misconception is that if a server only accepts server-generated session identifiers, it is safe from fixation. This is false.

2. Inspect 5 websites for which you got login credentials

   - Do these website use HSTS (HTTP Strict Transport Security)?
   - Does the session ID cookie of these websites use *Secure* and/or *HttpOnly*?

Make a table of these websites and their support for HSTS, the name of the cookie and the usage of *Secure* and *HttpOnly* for that cookie.
Solution:

Website	**HSTS**	**Cookie Name**	**Secure**	**HttpOnly**
facebook.com	yes	multiple cookies (c_user, xs, sb)	yes, yes, yes	no, yes, yes
mail.ru.nl	no	cadata	yes	yes
linkedin.com	no	li_at	yes	yes
github.com	yes	_gh_sess	yes	yes
gmx.net	yes	905a7a5991a00898953878290a55d118	yes	yes

Table 1: Website Information