Name:Ahmed Almass

Lab 4: Attacking Session Management

Web Application Security

Table of Contents

[Lab Outcome 2](#_Toc524601642)

[Background Reading 2](#_Toc524601643)

[Required Hardware/Software 2](#_Toc524601644)

[Introduction 2](#_Toc524601645)

[1.0 Spoofing 2](#_Toc524601646)

[2.0 Session Hijacking 2](#_Toc524601647)

[3.0 Session Fixation 2](#_Toc524601648)

[4.0 Sign-Off – Lab 4: Attacking Session Management 3](#_Toc524601649)

Web Application Security

Lab 4: Attacking Session Management

# Lab Outcome

Exploit session management controls.

Background Reading

Read the textbook sections listed in the Course Schedule.

Required Hardware/Software

* WebGoat v7.1
* Burp

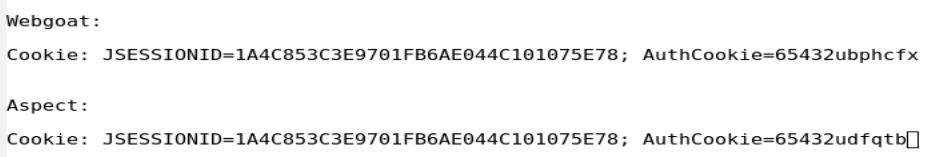
# Introduction

Session management controls what users can do and how they can interact with the system. It is crucial in maintaining a user’s identity across multiple requests.

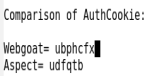
# 1.0 Spoofing

In WebGoat, complete the **Session Management Flaws > Spoof an Authentication Cookie** lesson.

When intercepting the logins i was able to capture the cookies.



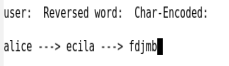
When comparing the cookie the only difference was the last bit of the authcookie.



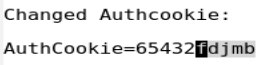
The last bit uses char encoding which is “transforms data into another format using a scheme that is publicly available so that it can easily be reversed”. By using a site [<http://yehg.net/encoding/?i=Encode+This+String>] to encrpt it. By looking at the Decrypted value, the value is reversed username.



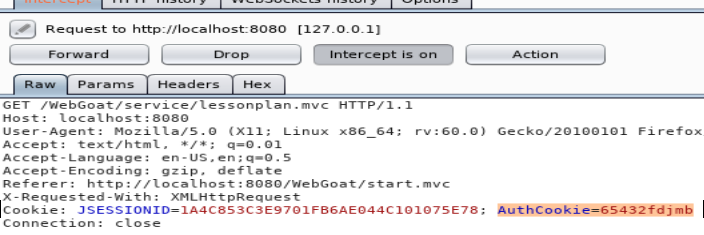
To login as alice you’ll have to reverse the word and Char-Encoded it.



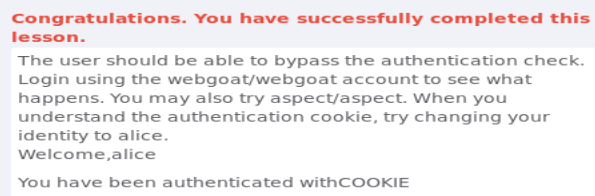
Than change the last bit of the authcookie as shown.



Aftwards I sign in to a regular user, intercept it with burp and replace the whole cookie as shown.



Success login



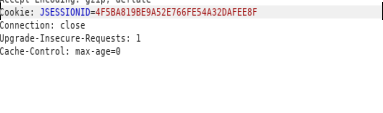
# 2.0 Session Hijacking

In WebGoat, complete the **Session Management Flaws > Hijack a Session** lesson.

1st. Refresh the page & intercepting it with burp. WEAKID is the session cookie.



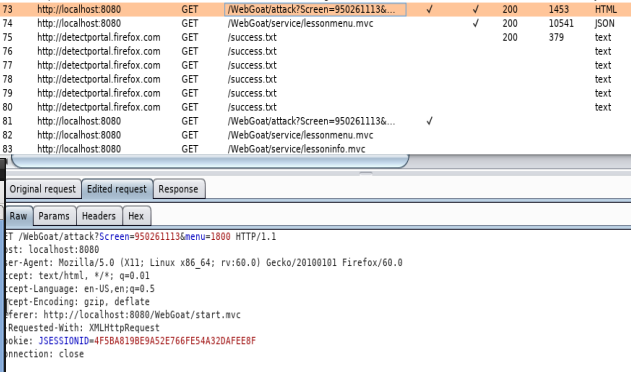
Removing the cookie, so it forces the application to make a new cookie



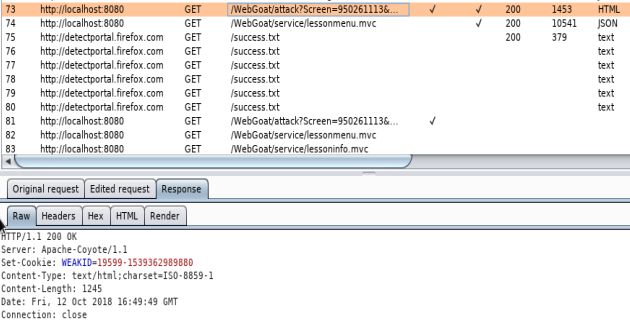
Going to HTTP history check the response of the application for the new cookie.

Orginal request cookie:

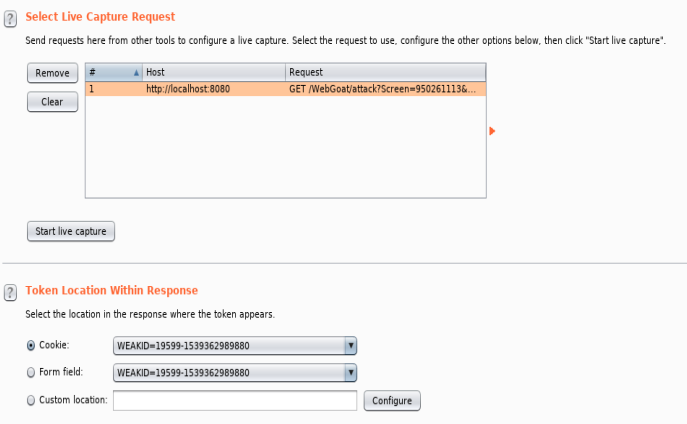
# 

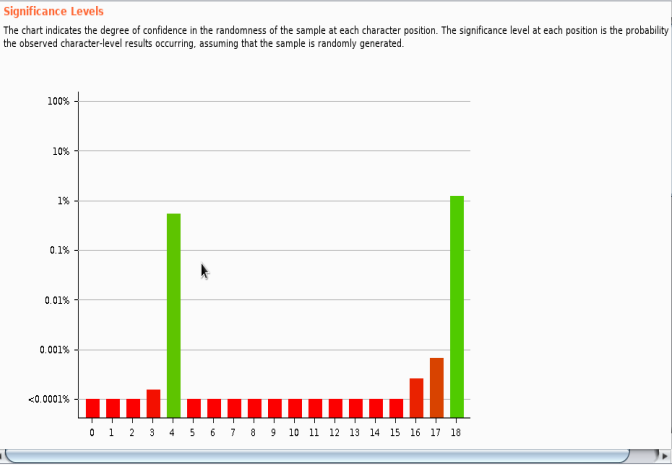
Edited:

Response (new cookie):

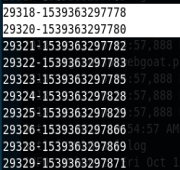


Now we generate the cookie to see if there is a patern.

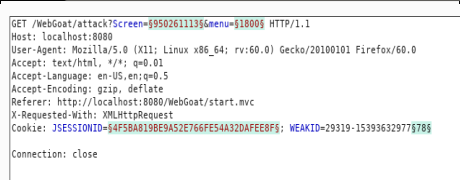




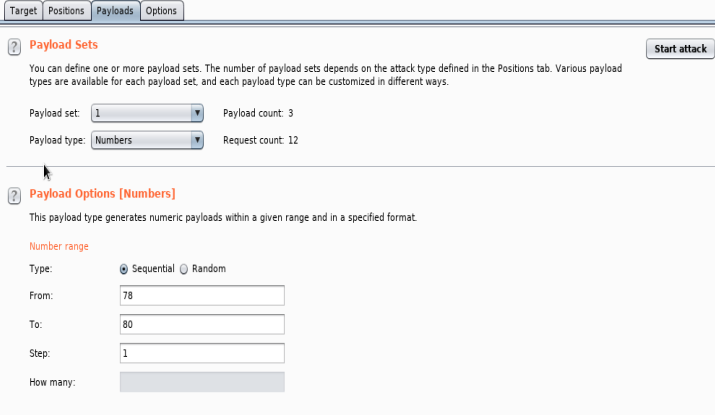
When checking the tokens you’ll see the patern, if there a gap its possible for another person session.



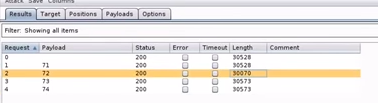
After you’ll use burp to generate cookie session. Placeing a value inbetween the gap [example the numbers above picture 29319, also placeing the last 2 numbers as a varaible]



Sequence the values by 1.

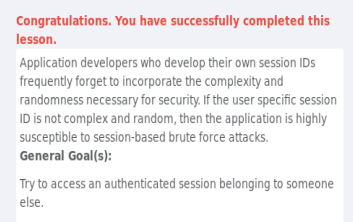


The results will show the status and the one that has the different length will authentication session that belongs to someone else.



Response



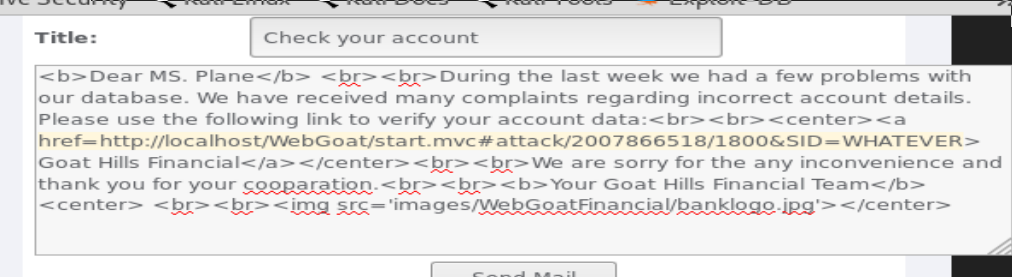


# 3.0 Session Fixation

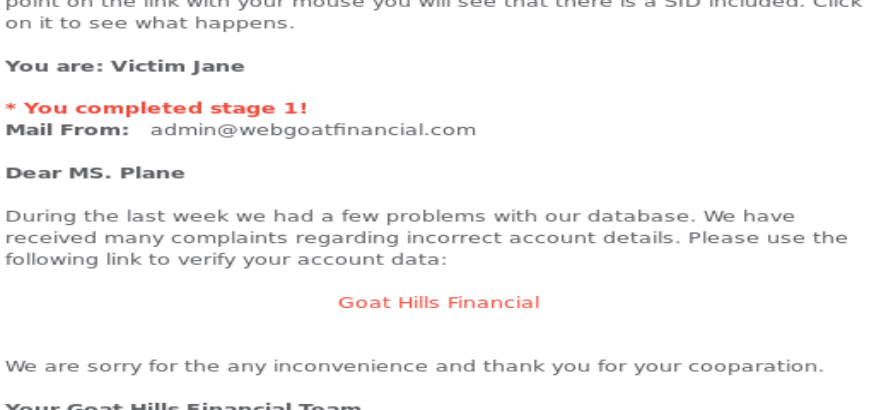
In WebGoat, complete the **Session Management Flaws > Session Fixation** lesson.

1st Stage:

Send him an email with a link that is custom session link.



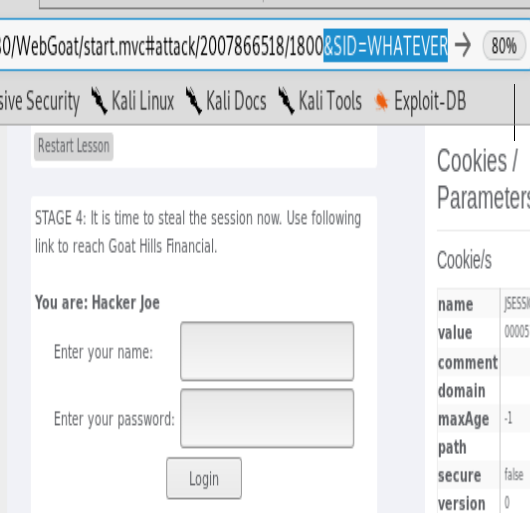
2nd Stage: The victim will receve a an email with a link.



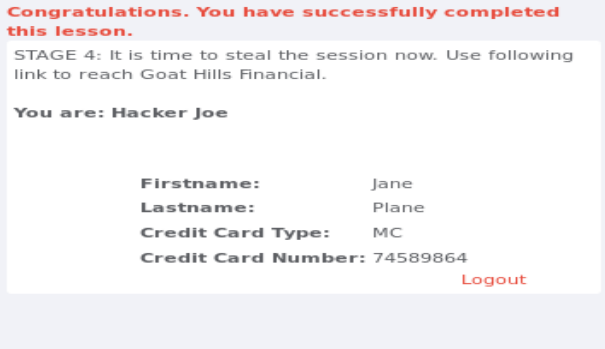
3rd Stage: victim Clicks link and tries to access a login page.



4th: Hacker will change the URL to the custom SID created for the victim.



Success!



# 4.0 Sign-Off – Lab 4: Attacking Session Management

Detach this page and submit it to your instructor to indicate that you have completed each section.

Name:

Student ID:

|  |  |
| --- | --- |
| **Section** | **Instructor Initials** |
| 1.0 Spoofing |  |
| 2.0 Session Hijacking |  |
| 3.0 Session Fixation |  |