COMP 4632
Practicing Cybersecurity: Attacks and Counter-measures

Week 9 Lab Exercise
Topic: *Web Application Vulnerabilities*

Lab Objective

In this lab, you will try to perform web application attacks. This will include simulated teaching lab and custom made application and aim at achieving the following objectives:

- Understand web application authentication and session management
- Understand basics of web sessions, cookies and authentication methods
- Identify and exploit web vulnerabilities

Task 1 – Playing with Browser Cookies

Browser cookies are commonly used by web applications to stored information at client side or maintain sessions. In this task, you will work on a few exercises to be familiar with the characteristics of browser cookies.

Task 1.1 Adding Browser Cookies

- Start Fiddler on the Windows 7 VM
- Access the course website (https://course.cse.ust.hk/comp4632/) from the Chrome browser in Windows 7 VM
- Verify that the traffic is passing through Fiddler
- Check the traffic when browsing the course web site. There should be no cookies sent. If there are any, clear the cookies in browser and restart this task.
- Do not close the browser or clear browser cookies in this task unless you are explicitly told to do so.
- Try to understand the differences between the 3 different methods to add browser cookies.

Task 1.1.1 Adding cookies in HTTP request headers

- Browse to the testing page https://course.cse.ust.hk/comp4632/lab9_test/cookies.php which shows the cookies sent in the request in an Array object (it should be empty).
- Turn on intercept request mode in Fiddle. Reload the testing page. In the intercepted request, add the following HTTP header:

```
Cookie: testa=1
```

- See the output of the testing page. Compare the result with Chrome DevTools (Press F12, then "Resources" panel > "Cookies" > "course.cse.ust.hk"). Are they consistent?
- Disable interception mode in Fiddler, and reload the testing page. Observe the cookie(s) in HTTP request sent, compare that with the output of the testing page, and Chrome DevTools. Are they consistent now?

Task 1.1.2 Adding cookies with HTTP response headers

- Turn on intercept response mode in Fiddler, browse the testing page again. In the intercepted request, add the following HTTP header.

```
Set-Cookie: testb=2
```

- See the output of the testing page. Compare the result with Chrome DevTools. Are they consistent?
- Disable interception mode in Fiddler, and reload the testing page. Observe the cookie(s) in HTTP request sent, compare that with the output of the testing page, and Chrome DevTools. Are they consistent now?

Task 1.1.3 Adding cookies with Javascripts

- Browse to the testing page. Go to the “Console” panel in Chrome DevTools. Type the following and press <ENTER>:
```
document.cookie=”testc=3”
```
- See the output of the testing page. Compare the result with Chrome DevTools. Are they consistent?
- Reload the testing page. Observe the cookie(s) in HTTP request sent, compare that with the output of the testing page, and Chrome DevTools. Are they consistent now?
- Keep the testing page opened. Type the following in the DevTools console and press <ENTER>
```
document.cookie=”testd=4”
```
- Reload the testing page. Observe the cookie(s) in HTTP request sent, compare that with the output of the testing page, and Chrome DevTools. Do you still see the testc cookie?

Task 1.2 Understanding Characteristics of Browser Cookies

- Keep the browser opened at the testing page. Set the following cookies (each line represents one cookie) either with HTTP response headers or Javascript.
```
teste=5; domain=.cse.ust.hk
testf=6; path=/comp4632/lab9_test
testg=7; domain=.cse.ust.hk; path=/comp4632/lab9_test
testh=8; Expires= Fri, 01 Jan 2016 00:00:00 GMT
testi=9; HttpOnly
testj=10; Secure
```
- Check the cookies on this page via the following 3 methods
  - “Cookie” header in the corresponding HTTP request (check in Fiddler)
  - Chrome DevTools “Resources” panel > “Cookies”
  - Chrome DevTools “Console” panel, type the following and press <ENTER>:
    ```
    document.cookie
    ```
- Perform the following actions. After <u>each</u> action, check what cookies are set, and observe the differences between the 3 different methods for checking cookies
  i. Browse to https://www.cse.ust.hk/
  ii. Browse to https://course.cse.ust.hk/comp4632/
  iii. Browse to https://www.cse.ust.hk/comp4632/lab9_test/not_exist.html (yes, you will see an error page)
  iv. Browser to the page http://course.cse.ust.hk/comp4632/lab9_test/cookies.php (note that the protocol is HTTP instead of HTTPS)
  v. Close the browser and reopen the testing page again https://course.cse.ust.hk/comp4632/lab9_test/cookies.php.

Task 2 – Web Application Vulnerabilities – Authentication and Session Management Flaws

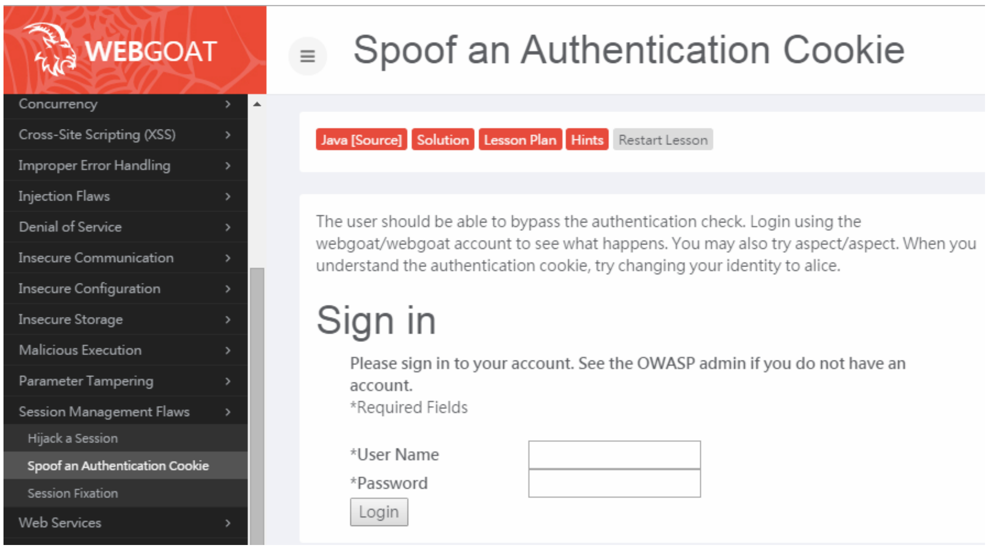
In this task, we will walk through some of the common application vulnerabilities related to authentication and session management. You are expected to understand the how to identify and exploit vulnerabilities in the WebGoat lessons with Fiddler.

Task 2.1 Session Management Flaws

- Start Fiddler on the Windows 7 VM
- Access the WebGoat web interface from the Windows 7 VM
- Login with the Webgoat User account
  - Username: guest
  - Password : guest
- Verify that your WebGoat traffic is passing through Fiddler

Task 2.1.1 Spoof an Authentication Cookie

- WebGoat Lesson – Spoof an Authentication Cookie
- On the WebGoat menu, select Session Management Flaws → Spoof an Authentication Cookie



The user should be able to bypass the authentication check. Login using the webgoat/webgoat account to see what happens. You may also try aspect/aspect. When you understand the authentication cookie, try changing your identity to alice.

Please sign in to your account. See the OWASP admin if you do not have an account.
*Required Fields

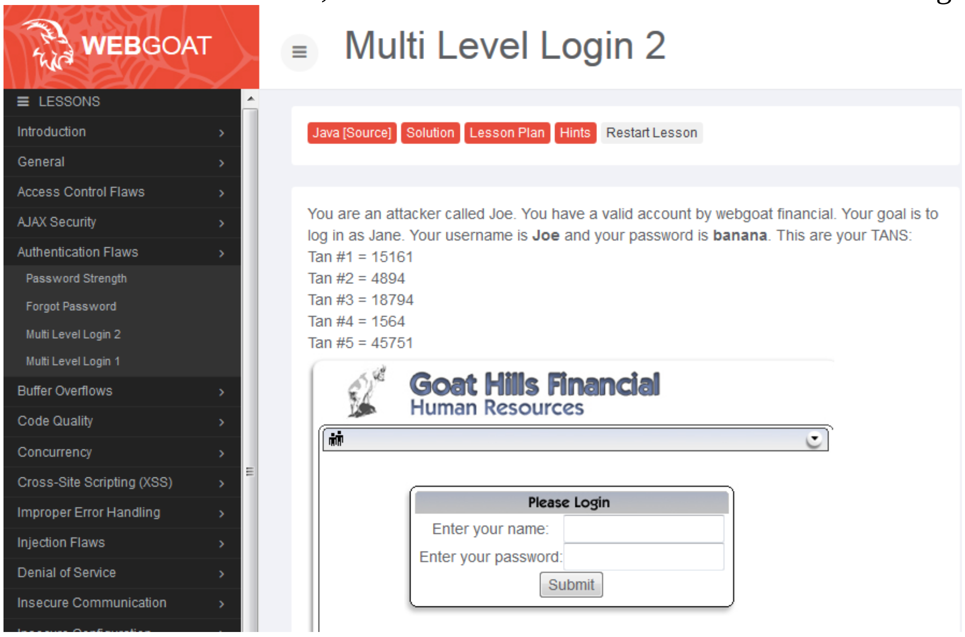
- Try to login using the provided accounts pairs for multiple time, “webgoat:webgoat” and “aspect:aspect”.
- Observe the pattern of the assigned session cookie values for different login.
- Observe and guess how the cookie values are generated cookie values on the server side.
- By an educated guess, try to create a session cookie which let the server thinks you are authenticated as “alice”. Inject this cookie into your browser and complete the lesson.

Task 2.2 Authentication Flaws

- More applications adopt multi-factor or multi-steps authentication nowadays, but improper implementation may lead to vulnerabilities, such as impersonation, account hijack, privilege escalation, e.t.c.

Task 2.2.1 Flaws in Multi-steps Login Implementation

- WebGoat Lesson – Multi Level Login 2
- On the WebGoat menu, select Authentication Flaws → Multi Level Login 2



You are an attacker called Joe. You have a valid account by webgoat financial. Your goal is to log in as Jane. Your username is **Joe** and your password is **banana**. This are your TANS:
Tan #1 = 15161
Tan #2 = 4894
Tan #3 = 18794
Tan #4 = 1564
Tan #5 = 45751

- Try to login using the provided account "Joe:banana" and submit a correct TANS pair and observe the 2 HTTP requests sent.
- Usually application regard a user as an authenticated user after the first request, but this intermediate authentication state may cause many problems.
- In some applications, you may also be able to bypass the second step of authentication completely.
- Try to modify the request and login as the user "Jane" without knowing her password or TAN.

End of Lab