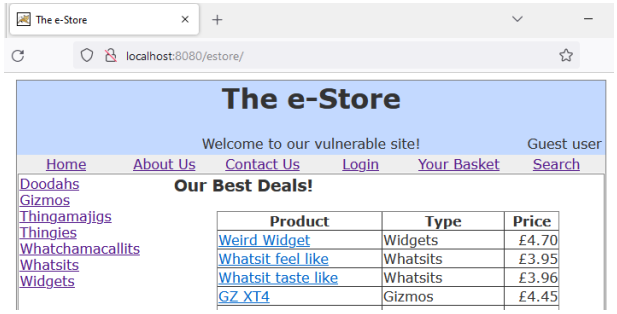
Coursework

This coursework aims to consolidate your learning of web security. It relates to a vulnerable fictitious website (e-Store). You will have the opportunity to both attack then defend this web app.

In your Windows VM, open the XAMPP Control Panel and start the **Tomcat** service. Access the e-Store through the URL: http://localhost:8080/estore/



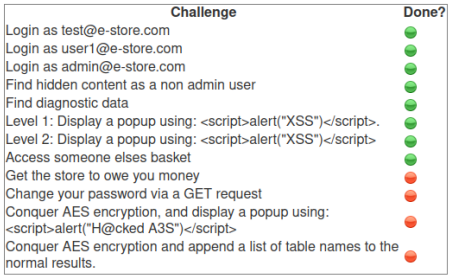
Notes:

1. Because Tomcat uses port 8080, you may want to set the port for Burp to some other value, for example 8082. Consequently, to use Burp alongside your browser, your browser's proxy should be set to use 127.0.0.1 with port 8082.
2. You do **not** need to start the Apache and MySQL services in this coursework.

Task(s)

Task 1: Offensive Security

The aim of this task is to exploit certain vulnerabilities on the website.
The website has a score board that contains a list of 12 challenges. The score board can be accessed by browsing to the "**About Us**" page then clicking on the "**Scoring Page**" link or by directly accessing: http://localhost:8080/estore/score.jsp
Challenges will be shown as green once solved as shown below:



Notes:

1. When you stop the Tomcat service and access the e-Store again, the score board will be reset and will show red against all challenges. This is normal and is nothing to worry about.
2. You do not need to attempt the challenges in the order specified in the score board.
3. **You do not need to solve challenge 2** (login as user1) **nor challenge 3** (login as admin) because they are similar to challenge 1.
4. I will provide you with the solution to **challenge 5** (Find diagnostic data) because it will help you with other challenges. The e-Store uses a parameter called *debug* to display error messages. For example, visiting the following page will solve challenge 5: http://localhost:8080/estore/login.jsp?debug=true
5. To complete the ***9 remaining challenges***, you mainly need a browser, the browser's web development tools, and Burp Suite. But feel free to use any other tool you deem appropriate (including those in the Kali VM).
6. Remember that, in this task, you are taking the perspective of an attacker. Therefore, you shouldn't attempt to edit and modify the website's source code (on the server side) to complete the challenges. Here are some hints to help you complete the challenges:

Challenge	Hint
Challenge 1: Login as test…	SQL-injection. Your injected code needs to terminate the closing bracket shown in the error code. Remember to add the ?debug=true to the URL.
Challenge 4: Find hidden content	
Challenge 6: Level 1 XSS	Reflective XSS.
Challenge 7: Level 2 XSS	Stored XSS.
Challenge 8: Access someone…	
Challenge 9: Get the store…	Time for a bit of shopping and some simple arithmetic.
Challenge 10: Change your password…	From POST to GET.
Challenge 11: Conquer AES encryption, and display a popup	Inject script in the "Type" input box of "Advance Search". The form has a JavaScript function that calls another function which replaces special characters with their URL encoding. Try to bypass it. To bypass any JavaScript code (especially when using Firefox), editing the code (as we did in the lab when editing HTML) will not make changes you make persist. You can do a bit of research online to see how to ensure your changes can be made to last, but here is a summary of what you will find: take that JavaScript code to notepad, make all the necessary changes, copy this new version of the JavaScript function and paste it in the "Console" (of the browser's web developer's tools), then hit enter. This will result in the new JavaScript code taking effect, allowing you to bypass any validation taking place and stopping you from successfully completing the attack. This is just one way of completing the challenge. Another way could be to use Burp to intercept the "response", not the request. And yet there is another third possible way, manipulating the encrypted q variable.
Challenge 12: Conquer AES encryption and append a list of the table names	Combine the bypassing of validation from the previous challenge with SQL injection. Remember to use *?debug=true* to get the website to display errors. You need to work out what database the web app is using (not just assume it is MySQL). You can do that by searching online for the error message thrown by the app while attempting the first challenge, or scanning the app, etc. This hint is to tell you that the database system used is HSQLDB. By searching online, you will find that one way to list tables from the data dictionary is to query from information_schema.system_tables

Task 2: Defensive Security

The e-Store website is available under the folder **c:\xampp\tomcat\webapps\estore**

Implement **Five (5)** security measures to mitigate against some of the e-Store vulnerabilities including:

- **Three (3)** security measures related to the configuration of Tomcat. For this, search online for how to strengthen the security of a Tomcat server and use your findings to check whether those security controls are found in your Tomcat configuration files.
- **Two (2)** security measures to address vulnerabilities in the source code (jsp files) of the website:
    - one to mitigate against SQL injection and
    - one to mitigate against XSS.