Cibersegurança – Módulo 2

Segundo trabalho prático

The group should have:

- A project on Google Cloud Platform (GCP)
- A GitHub repository within the “isel-deetc-computersecurity” organization

Delivery:

- Report with evidence and / or explanations requested in each question, the name of the virtual machine created in the GCP and the port used to host the Juice Shop application.

Pretende-se analisar fazer análise estática do código e análise dinâmica à aplicação Juice Shop, uma aplicação web com erros propositados com o objectivo de ganhar maior conhecimento sobre vulnerabilidades em geral e em aplicações web em particular.

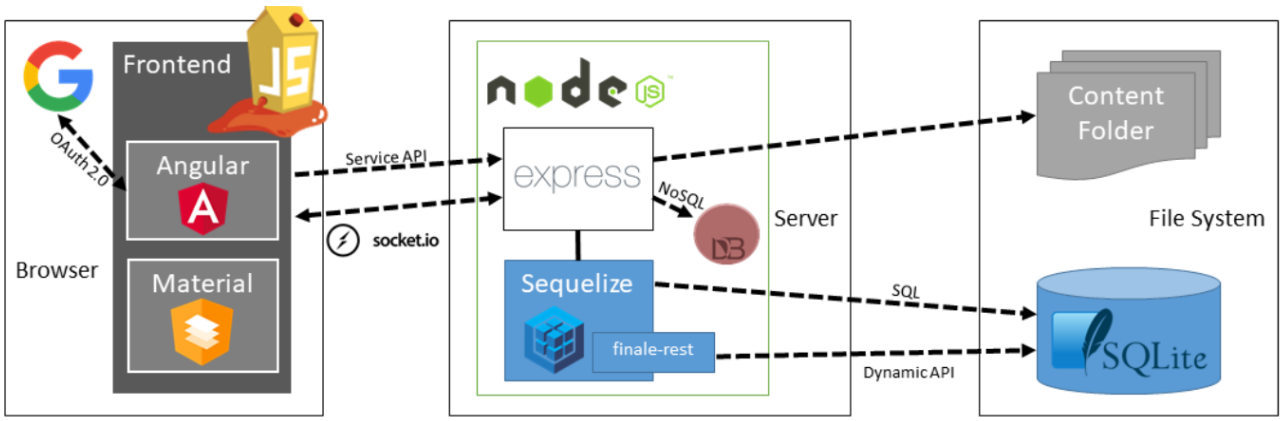

Figura 1: Arquitetura da Juice Shop

The Juice Shop architecture includes a frontend developed in Angular, a backend developed in node.JS and the use of relational and non-relational databases, as illustrated in Figure 1.

<u>Análise estática</u>

1. Get the source code for the Juice Shop app, https://github.com/juice-shop/juice-shop and add the code to your group's GitHub repository.

   A message was sent by Moodle with a link to create a public repository within the organization “isel-deetc-computersecurity”.

2. Configure a GitHub Action to run CodeQL platform CWE analytics when there is a push event in the repository created in point 1.

3. Consider the CWE-89 vulnerability “Improper Neutralization of Special Elements used in an SQL Command” (https://cwe.mitre.org/data/definitions/89.html). Look for the entry “Database query built from user-controlled sources” under the routes/login.ts file in the list of vulnerabilities detected by Github Action CodeQL. Justify why CodeQL identified this code as vulnerable. Include *source* and destination (*sink*) information. (https://owasp.org/www-community/attacks/SQL_Injection).

4. Static code analysis can have errors such as false positives or false negatives. In the context of the previous example, where a possible SQL Injection vulnerability was detected, give examples of what could be a false positive and false negative.

5. Consider the following documentation on CodeQL and the ability to do flow analysis in source code in Java or javascript (https://codeql.github.com/docs/codeql-language-guides/analyzing-data-flow-in-java/#analyzing-data-flow-in-java) (https://codeql.github.com/docs/codeql-language-guides/analyzing-data-flow-in-javascript-and-typescript/#analyzing-data-flow-in-javascript-and-typescript). The analysis performed by CodeQL can be local or global. The QL interrogation that detected the vulnerability of point (3) does which of these types of analyses?

Dynamic analysis and attack proxy

6. Host the application on the Cloud, using a VM from a GCP project, as explained here: https://github.com/juice-shop/juice-shop#google-compute-engine-instance

7. With the browser's programmer tools (F12 in chrome or firefox) discover the path that gives access to the ratings list - “score board”, present in one of the javascript files loaded by the frontend. Present the actions taken. This question corresponds to the “Find the carefully hidden 'Score Board' page” challenge: https://pwning.owasp-juice.shop/part2/score-board.html

8. Perform an SQL Injection attack on the Login form, targeting the admin user. Tutorial help available at this link: https://demo.owasp-juice.shop/#/hacking-instructor?challenge=Login%20Admin.

9. Install and verify the correct operation of the Zed Attack Proxy (ZAP) available here https://owasp.org/www-project-zap/:

a. In the “Quick start” option, choose the “Manual Explore” option, and indicate the site https://www.example.org, with the HUD active, choosing the Firefox browser (it may be necessary to install the browser). Check you can access the site and the ZAP commands in the same window.
b. Access the Juice Shop application through the browser launched in the previous point.

10. Using the ZAP tool, find out the password of the admin user admin@juice-sh.op. The password starts with “admin” and has a 3-digit numeric suffix. Show how you can by fuzzing find the correct password.

11. Look for products with the word “Lemon”. Notice the address bar. Try changing the search criteria to “Lemon1” directly in the address bar. On the results page where is the search criteria text entered?

12. Solve the “DOM XSS” challenge by injecting the text <iframe src="javascript:alert(`xss`)"> so that the browser has to render a page with the injected text (https://pwning.owasp-juice.shop/part2/xss.html#perform-a-dom-xss-attack)

13. [extra] Show how to solve the “Post some feedback in another users name” challenge using the ZAP proxy.

14. [*extra*] Show how, through a Cross-site scripting (XSS) and social engineering attack, an attacker can obtain the cookie named “token” stored in the victim's browser.