# 02233 Network Security
# Red day

16/04/2024

In this lab, we will play the OWASP Juice Shop web application (GitHub link). This application contains many vulnerabilities/challenges that can be seen here alongside hints and solutions (in case you are stuck). You can find the list of challenges we have selected for this lab in Table 1. The challenges are sorted in ascending order in terms of difficulty, and provide hints to get you started (the names are also hints). Since the application is written in Angular (a client-side rendering JS framework for SPA applications), we recommend you start by opening the site in the burp browser and inspect the "`main.js`" file, which contains very valuable information such as endpoints, paths and many other interesting details.

| Name | Description | Hint |
|---|---|---|
| Score Board | Find the carefully hidden "Score Board" page. | link |
| DOM XSS | Perform a DOM XSS attack with `<iframe src="javascript:alert(`xss`)">` | link |
| Zero Stars | Give a devastating zero-star feedback to the store. | link |
| Admin Registration | Register as a user with administrator privileges. | link |
| Admin Section | Access the administration section of the store. | link |
| Deprecated Interface | Use a deprecated B2B interface that was not properly shut down. | link |
| XXE Data Access | Retrieve the content of `C:/Windows/system.ini` or `/etc/passwd` from the server. | link |
| Database Schema | Exfiltrate the entire DB schema definition via SQL Injection. | link |
| User Credentials | Retrieve a list of all user credentials via SQL Injection | link |
| Outdated Allowlist | Let us redirect you to one of our crypto currency addresses which are not promoted any longer. | link |
| Local File Read | Gain read access to an arbitrary local file on the web server. | link |
| Bjoern's Favorite Pet | Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the truthful answer to his security question. | link |
| API-only XSS | Perform a persisted XSS attack with `<iframe src="javascript:alert(`xss`)">` without using the frontend application at all. | link |
| Cross-site Imaging | Stick cute cross-domain kittens all over our delivery boxes. | link |
| Login Bjoern | Log in with Bjoern's Gmail account **without** previously changing his password, applying SQL Injection, or hacking his Google account. | link |
| Weird Crypto | Inform the shop about an algorithm or library it should definitely not use the way it does. | link |
| Vulnerable library | Inform the shop about a vulnerable library it is using *(Mention the exact library name and version in your comment).* | link |
| XXE DoS | Give the server something to chew on for quite a while. | link |

Table 1: List of challenges

You can find the installation guide on GitHub, to choose from either Docker or Vagrant. *Note: On Windows, you may need to add an environment variable to the Docker container to complete some of the challenges. If you can't pull the*

*image try the Vagrant version.* We recommend using Burp Suite (Community) to complete the challenges we have gathered here. Burp is a Swish-knife for web applications penetration testing. This tool is better known for its capabilities for manipulating communications between client and server. In addition, it contains very useful tools for fuzzing and web mapping (not scraping, and not crawling, only mapping through passive scanning).

Here are some examples of exploitation techniques that you will see today.

**Injection**

- **SQL Injection:** This method relies on sending SQL commands to the backend of the application. Invicti (another tool similar to Burp for enterprise web applications security assessment) has gathered a very neat SQL Injection cheat sheet here.
- **Bypassing Sanitation:** Similarly to SQL injections, not sanitised and validated inputs allow users to insert malicious payloads in the database and run random code in our servers. For example, users could store malware in the database as a blog comment, which is then delivered to any user requesting the page (e.g., a BeEF hook).

**URL manipulation**

- **Null Byte Poisoning:** This has to do with sanitation and the C language. Null byte means that we can introduce a null value as part of a string that will be thrown away when parsed. This is typically implemented in the hex form 0x00 or `%00`, or URL encoded `%2500`. E.g., `http://juice-shop.local/document.log%2500.md`
- **Path Traversal:** Some paths are links to files. Therefore, if the path is not sanitised properly, an attacker can traverse the file system to retrieve unauthorized data. E.g., `http://juice-shop.local/../../.ssh/id_rsa` may retrieve the private SSH key of the root user.
- **Robots & Security:** Very often we can find paths and other resources in "robots.txt" or "security.txt". These paths are meant for automated crawlers to know which resources should not be crawled (e.g., API endpoints, admin panels, and internal portals). In addition, this file may contain useful information, such as expected behavior patterns, account information, and more.

**APIs** API endpoints allow us to communicate with backends and maintain the state of the application. API endpoints with broken access control can be exploited to gain unauthorized access to resources and manipulate them. For example, users may be able to create new users and delete or retrieve information through HTTP requests via CORS methods (GET, POST, DELETE, and UPDATE).

**XSS** Cross-site scripting is a technique that forces a source (web application server) to load resources from a different source (another site). E.g., `htpp://juice-shop.local/?redirect=http://some-other-page.mal/`