# Exercise: Security Testing

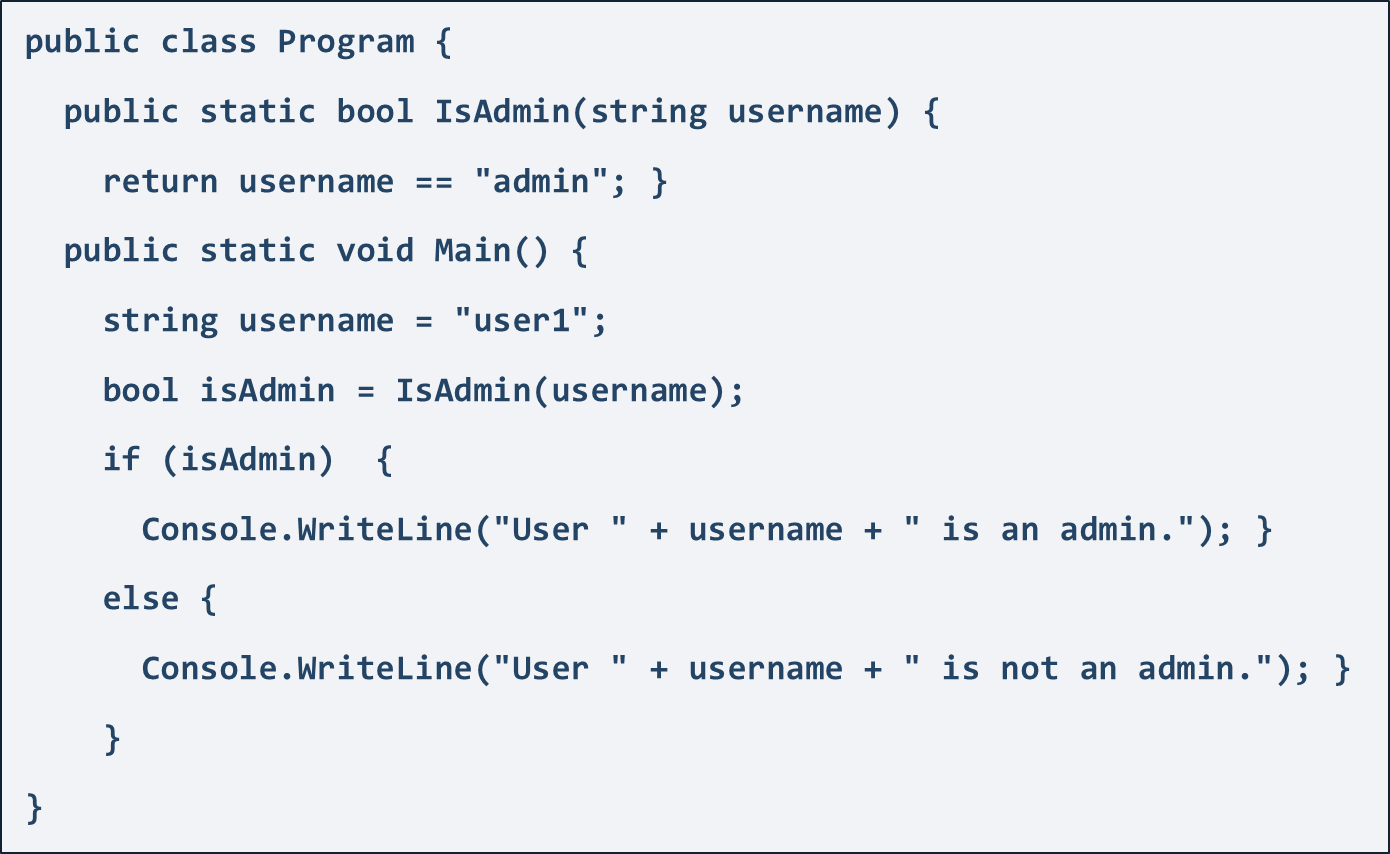


## Security Risks

### Broken Access Control

**What is Broken Access Control?**

* Bypassing security mechanisms that are meant to restrict access to certain resources or actions.
* Exploited to steal sensitive information, modify data, or cause other types of damage.
* **Example:**
  + The IsAdmin method checks if the username is equal to "admin".
  + No checking if the user is authorized to access the administrator status.



* **Exercise:** [**https://www.hacksplaining.com/lessons/broken-access-control/start**](https://www.hacksplaining.com/lessons/broken-access-control/start)
* **Broken Access Control Prevention**
  + Use role-based access control (RBAC) or attribute-based access control (ABAC)
  + Access control checks on both server-side and client-side
  + Use unique and unguessable resource identifiers
  + Use session management and limit cookies and tokens
  + Validate and sanitize input data
  + Use encryption, rate limiting, logging, and monitoring

### Unencrypted Communication

**What is Unencrypted Communication?**

* Refers to the transmission of sensitive information without any form of encryption or protection
* Occurs when using protocols such as HTTP
* Interception and eavesdropping on unencrypted communication using tools such as packet sniffers
* Steal sensitive confidential data /usernames, passwords, credit card numbers/
* **Exercise:** [**https://www.hacksplaining.com/exercises/unencrypted-communication**](https://www.hacksplaining.com/exercises/unencrypted-communication)
* **Unencrypted Communication Prevention**
  + Use secure communication protocols (HTTPS, SFTP, FTPS, SMTPS)
  + Encrypt data in transit (SSL/TLS, digital certificates)
  + Encrypt sensitive data at rest (database, file-level, full-disk)
  + Use strong passwords, access controls, and 2FA
  + Implement network segmentation, firewalls, and IDS/IPS
  + Update and patch regularly

### SQL Injection

**What is SQL Injection?**

* Execution of malicious SQL queries in a web application's database
* Typically involves manipulating user input to inject SQL code into a query
* Common targets include login pages, search forms, and other forms that allow user input
* **Example:**
  + Original SQL Query:



* + Setting username to John & password to ' OR '1'= '1 produces



* + **The result:** If a user John exists – he is logged in without password
* **Exercise:** [**https://www.hacksplaining.com/exercises/sql-injection**](https://www.hacksplaining.com/exercises/sql-injection)
* **SQL Injection Prevention**
  + Use Prepared Statements
  + Validate all of the user information
  + Remove special characters from the user input
  + Never show SQL error messages to the user
  + Use different field names for user interface and database
  + Disable all unused features of the database
  + Limit user permissions for the database

### XSS - Cross-Site Scripting

**What is XSS?**

* Allows injecting client-side script into web pages viewed by other users
* The malicious code along with the original webpage gets displayed in the web client
* Allows hackers to gain greater access of that page
* Several types: including **Stored XSS**, **Reflected XSS**, and **DOM-based XSS**
* **Example: Simple Reflected XSS Attack**
  + Website with a search box for products. The search query is then reflected in the page URL:



* Malicious script entered into the search box:



* The resulting URL would look like this:

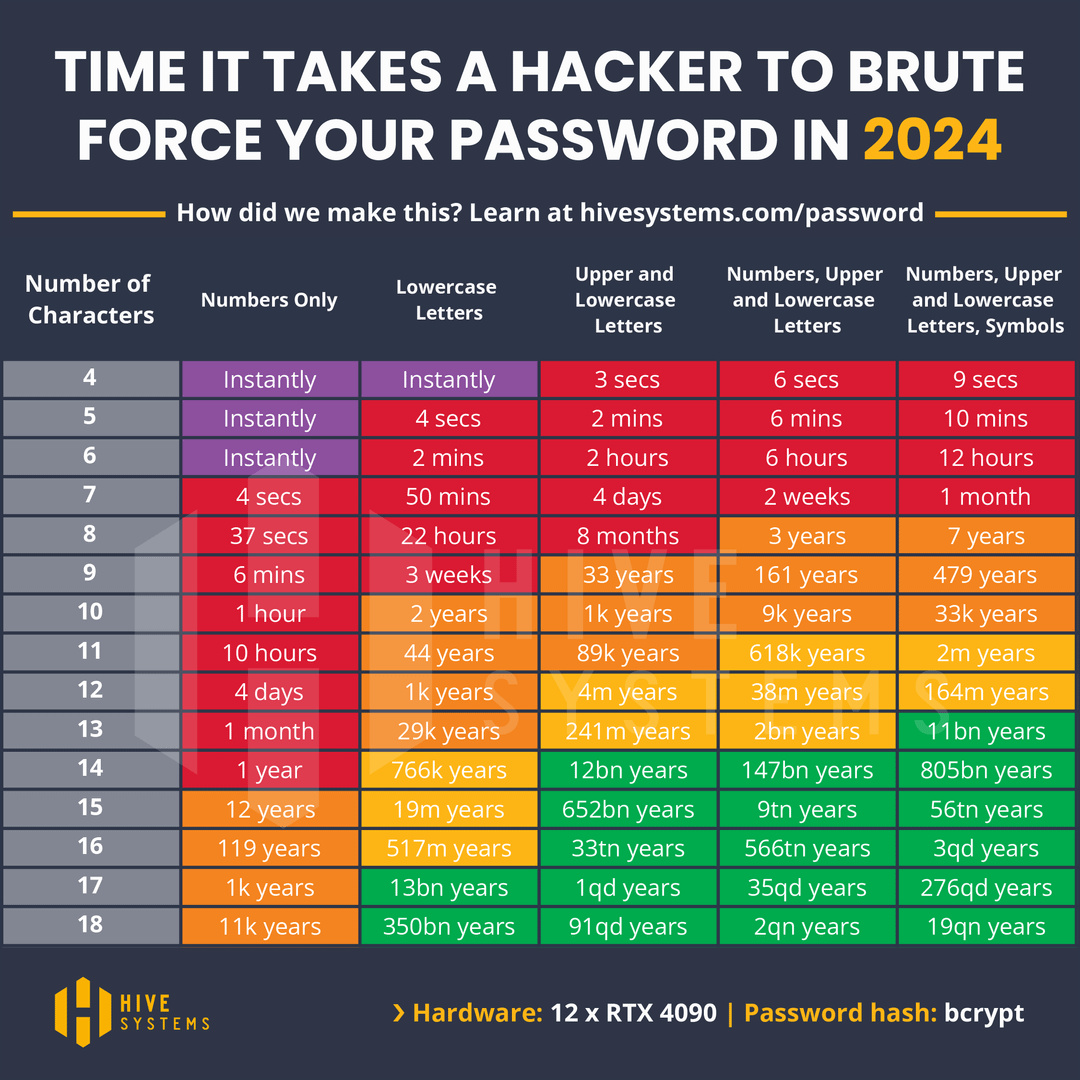


* When a user clicks, the script will be executed, causing an alert box to pop up with the message "XSS".
* **Exercise:** [**https://www.hacksplaining.com/exercises/xss-stored**](https://www.hacksplaining.com/exercises/xss-stored)
* **Preventing XSS**
  + Validate all input data from the user
  + Sanitize user input by encoding it before displaying it on the page or using it in a URL
  + Implement Content Security Policy (CSP) to restrict which scripts can be executed on the page
  + Use validating and escaping user-generated content
  + Never rely solely on client-side input validation

### Password Mismanagement

**What is Password Mismanagement?**

* Storing passwords in plaintext
* Not enforcing complexity requirements
* Not requiring password changes
* Not hashing passwords before storing them
* Allowing weak or common passwords that are easily guessed
* Transmitting passwords over an unencrypted channel
* Not having proper password recovery mechanisms
* **Example:**
  + An application has a password reset feature that sends a reset link to the user's email
  + The application does not properly verify the email address of the user before sending the reset link
  + An attacker submits a password reset request
  + The application sends the password reset link to the attacker's email address
  + The attacker uses the reset link to set a new password for the victim's account and takes over the account
* **Interesting Fact**

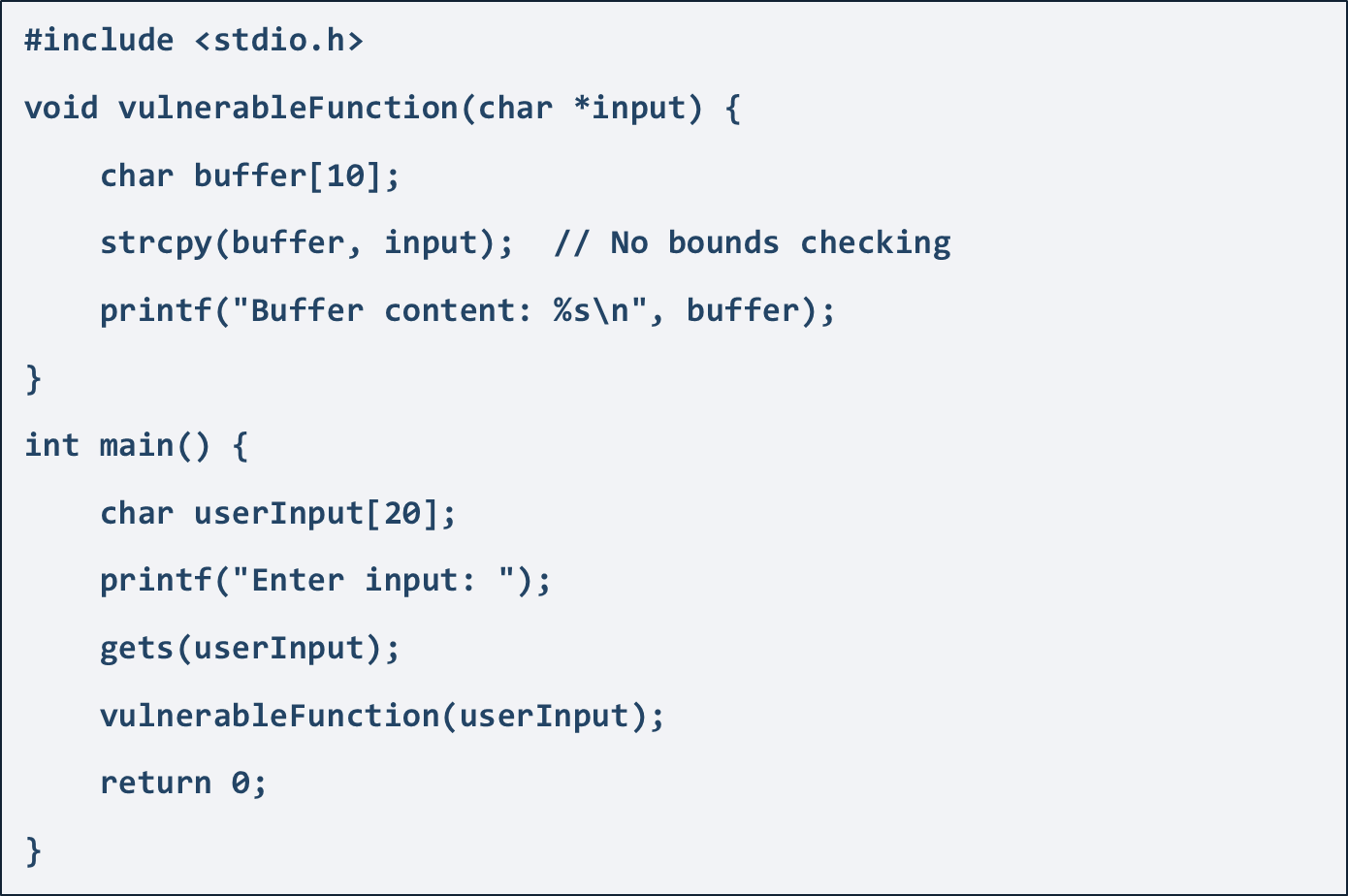


* **Exercise:** [**https://www.hacksplaining.com/exercises/password-mismanagement**](https://www.hacksplaining.com/exercises/password-mismanagement)
* **Password Mismanagement Prevention**
  + Use Third-Party Authentication if Possible
  + Ensure Password Complexity
  + Allow Password Resets via Email
  + Make sure reset links time out
  + Confirm Old Password on Reset
  + Prevent Brute-Forcing
  + Store Passwords with A Strong Hash, Salted
  + Timeout Sessions After Inactivity, and Provide a Logout Function
  + Use HTTPS for Secure Communication

### Buffer Overflow

**What is a Buffer Overflow?**

* Occurs when data exceeds the buffer's capacity, overwriting adjacent memory.
* Can lead to unexpected behavior, crashes, or security vulnerabilities.
* Affects both applications and system-level code.
* Common in C and C++, which do not manage memory automatically
* **Example: Simple Buffer Overflow Attack in C**
* **Application Accepts User Input:** The code uses gets() to read user input into the userInput array.
* **No Length Check:** The vulnerableFunction copies the input into the buffer array using strcpy(), which does not check if the input length exceeds the buffer's capacity.
* **Buffer Overflow:** If the input is longer than the buffer size (10 characters), strcpy() will write beyond the bounds of buffer.
* **Potential Consequences:** The overflow can overwrite adjacent memory locations, potentially altering data or program control flow, which can lead to arbitrary code execution or crashes.

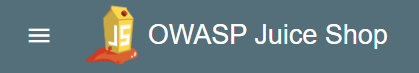


* **Exercise:** [**https://www.hacksplaining.com/lessons/buffer-overflows/start**](https://www.hacksplaining.com/lessons/buffer-overflows/start)
* **Preventing Buffer Overflow**
  + Validate all input data to ensure they fit within buffer limits.
  + Use safe programming practices and libraries that handle buffer sizes safely.
  + Apply compiler protections like stack canaries and non-executable stack segments.
  + Regularly update and patch software to mitigate known vulnerabilities.

### Other Risks

Learn more about other risks and vulnerabilities on [**https://www.hacksplaining.com/lessons**](https://www.hacksplaining.com/lessons)

## OWASP Juice Shop



OWASP Juice Shop is probably the most **modern and sophisticated insecure web application**! It can be used in security trainings, awareness demos, CTFs and as a guinea pig for security tools! Juice Shop encompasses vulnerabilities from the entire OWASP Top Ten along with many other security flaws found in real-world applications!

The application contains a vast number of hacking challenges of varying difficulty where the user is supposed to exploit the underlying vulnerabilities. The hacking progress is **tracked on a score board**. **Finding this score board** is actually one of the **(easy) challenges!**

### Setup

**From Sources**

* Install node.js
* Run **git clone https://github.com/juice-shop/juice-shop.git --depth 1** (or clone your own fork of the repository)
* Go into the cloned folder with **cd juice-shop**
* Run **npm install** (only has to be done before first start or when you change the source code)
* Run **npm start**
* Browse to **http://localhost:3000**

**Packaged Distributions**

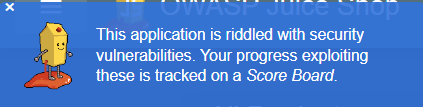
* Install a 64bit node.js on your Windows, MacOS or Linux machine
* Download **juice-shop-<version>\_<node-version>\_<os>\_x64.zip (or .tgz)** attached to   
  [latest release](https://github.com/juice-shop/juice-shop/releases)
* Unpack and **cd into the unpacked folder**
* Run **npm start**
* Browse to [**http://localhost:3000**](http://localhost:3000)

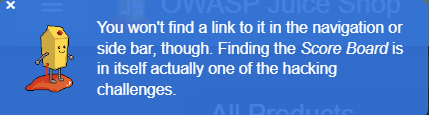


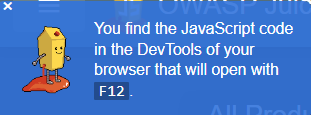
### Challenges

**Score-Board**

So, the first challenge is to find the Score Board





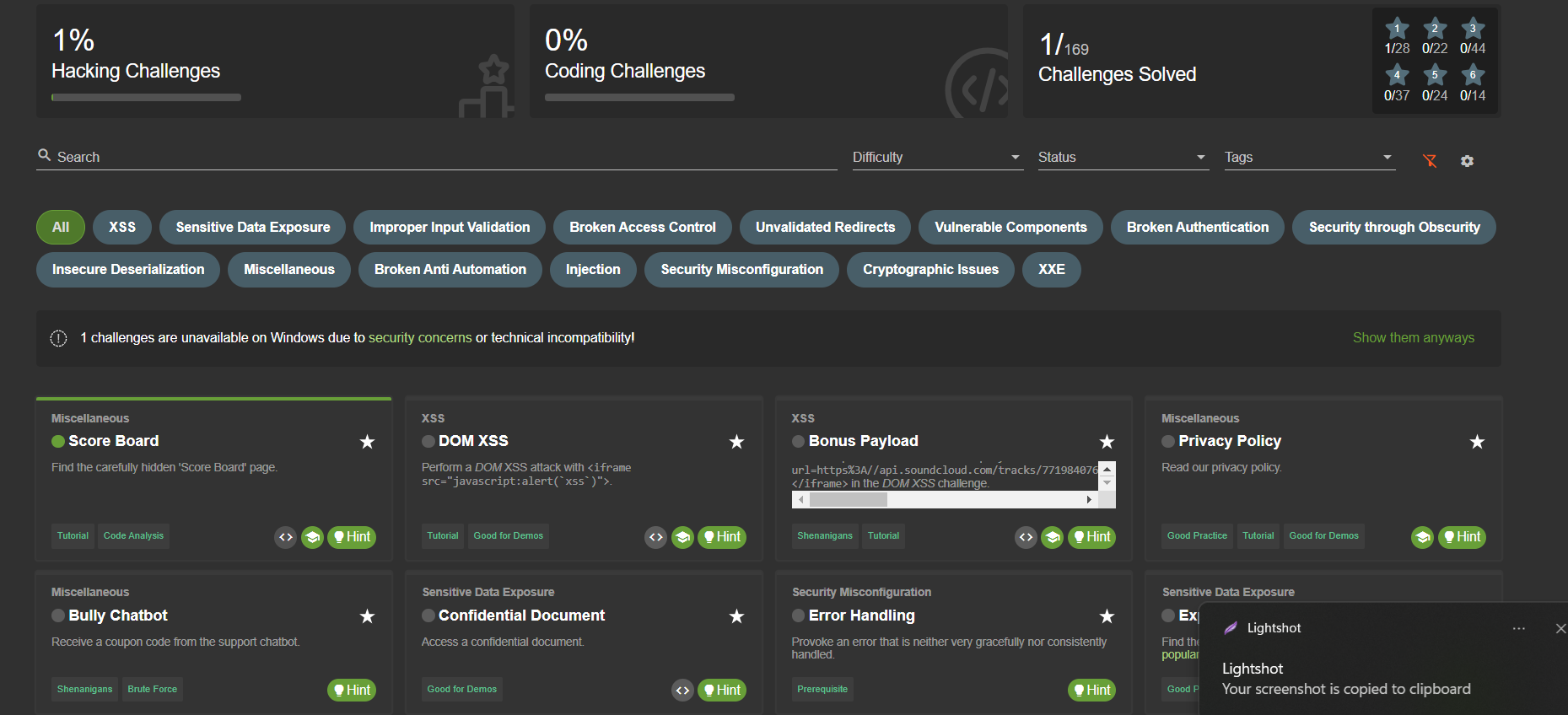


One of the simplest ways to begin analyzing a web application is to look at the HTML of a given page. To do so, simply use the "View Page Source" function of your web browser, usually available under the "Developer Tools" section or a similarly named menu.

While the HTML code itself doesn't present any attack opportunities, the list of JavaScript files referenced by the page within the "script" tags alerts one to some of the website's behind-the-scenes functions, some of which may prove to be a viable attack surface later.

* There is a URL that leads to the Score Board but it is not hyperlinked to.
* Knowing it exists, you can simply guess what URL the Score Board might have.
* Alternatively, you can try to find a reference or clue within the parts of the application that are not usually visible in the browser.

We hope that you tried to find the Score-Board by yourself, but if you're not <http://localhost:3000/#/score-board>



Each of the challenges has a description and also you can find hints and solutions all over the Internet, but try some for yourself.

## WebGoat: A deliberately insecure Web Application

**What is WebGoat?**

* Deliberately insecure web application, maintained by OWASP.
* Designed to teach web application security lessons.
* Demonstrates common server-side application flaws.
* The exercises are intended to be used to learn about application security and penetration testing techniques.

**WARNING 1:** While running this program your machine will be extremely vulnerable to attack. You should disconnect from the Internet while using this program. WebGoat's default configuration binds to localhost to minimize the exposure.

**WARNING 2:** This program is for educational purposes only. If you attempt these techniques without authorization, you are very likely to get caught. If you are caught engaging in unauthorized hacking, most companies will fire you. Claiming that you were doing security research will not work as that is the first thing that all hackers claim.

[**WebGoat on GitHub**](https://github.com/WebGoat/WebGoat?tab=readme-ov-file) **– Refer to the README file for a comprehensive guide and key details.**

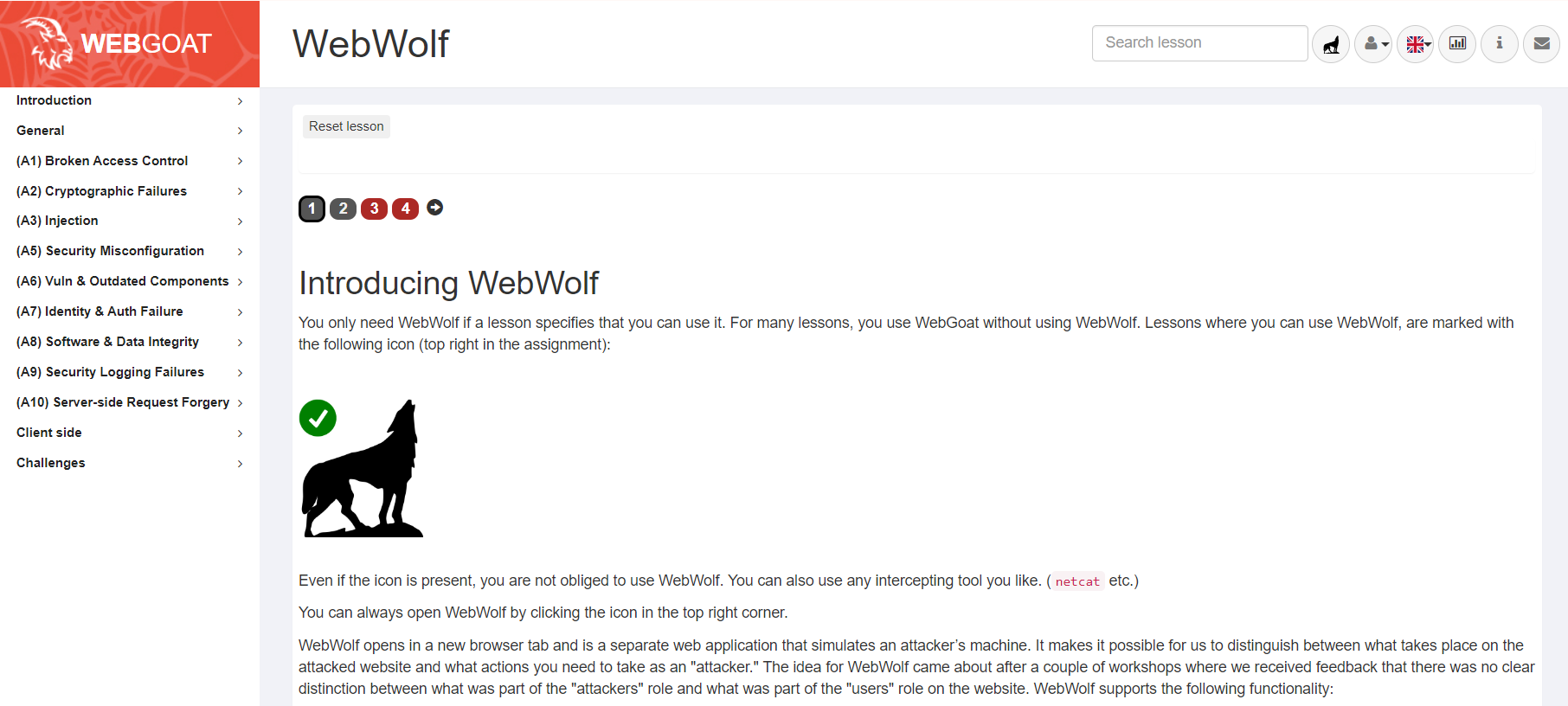
**Running the App:**

* The easiest way to run WebGoat is as a **Standalone**.
* Download the latest WebGoat release from <https://github.com/WebGoat/WebGoat/releases>
* Open your Command Prompt
* Navigate to where .jar file is downloaded and run the following commands:

**set TZ=Europe/Sofia *# or your timezone***

**java -Dfile.encoding=UTF-8 -jar webgoat-2023.8.jar**





## Learn Security Testing

Security and security testing are vital aspects of modern technology. As we delve into these topics, it's essential to understand that what we've covered so far is just the tip of the iceberg. The field of cybersecurity is vast and constantly evolving, requiring continuous learning and adaptation. Choosing a career in security means committing to a path of lifelong learning and vigilance.

As you progress in your studies and career, you'll encounter increasingly complex challenges and advanced techniques. Staying up-to-date with the latest threats and defenses is a must. To aid in your learning journey, here are some recommended platforms where you can practice and enhance your skills in security and security testing:

### [TryHackMe](https://tryhackme.com/)

**Online platform for learning and practicing cybersecurity skills**

* Variety of virtual environments and challenges that simulate real-world scenarios to solve
* Learning Paths
* Rooms
* Challenges
* Active Community

### [HackTheBox](https://www.hackthebox.com/)

* Platform that provides access to various vulnerable machines
* Regularly updated challenges and exercises covering a wide range of security topics
* Competitions for learning and practicing ethical hacking and penetration testing
* Active and supportive community of members sharing knowledge and strategies

### [PicoCTF](https://picoctf.org/)

* A cybersecurity competition that focuses on teaching and evaluating skills related to computer security
* Challenges that cover a wide range of security topics, including cryptography, binary exploitation, web security
* Challenges that are designed to teach specific concepts and skills, with detailed explanations and hints provided
* Free and open

### [WSTG Project - Web Security Testing Guide](https://owasp.org/www-project-web-security-testing-guide/stable/)

* Developed by the OWASP Foundation
* Detailed methodology and a set of testing techniques
* Four main categories
* Information Gathering
* Configuration and Deployment Management Testing
* Identity Management Testing
* Input Validation Testing