OWASP Juice Shop v14.3.1

Difficulty Level: 1-star (⭐) Walkthrough

Abhishek Sharma
https://www.linkedin.com/in/abhishek27sh/

Description

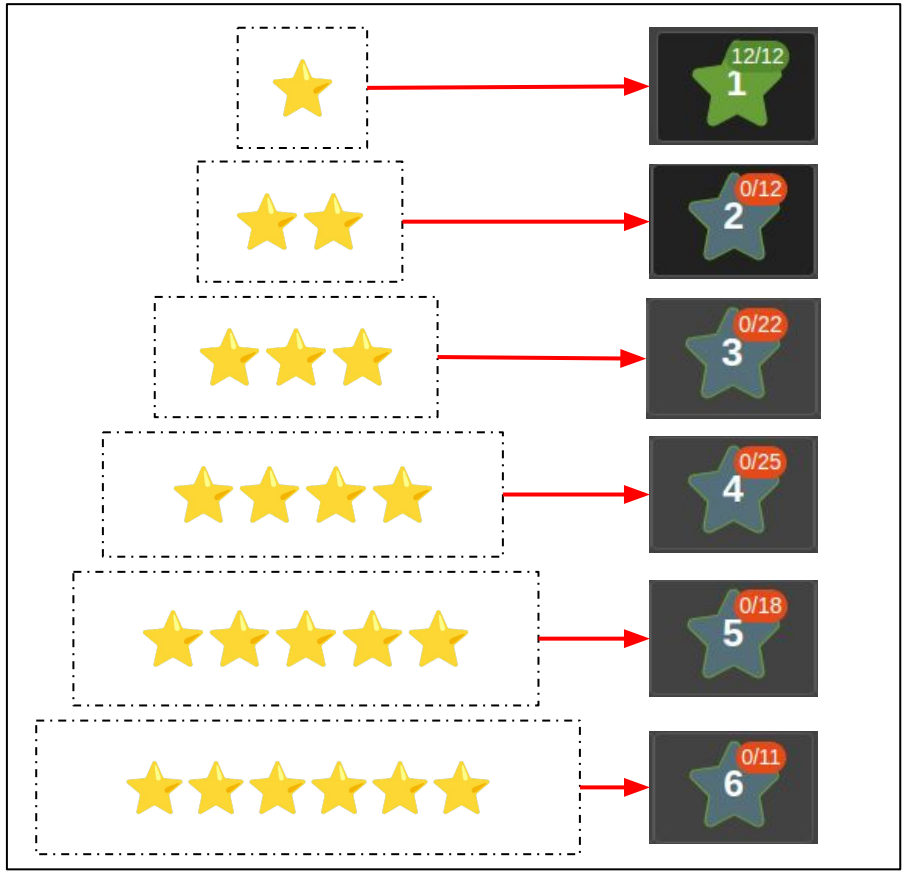
1. Insecure web application
2. Covers vulnerabilities from entire "Owasp Top 10"
3. Version v14.3.1 covers 6 difficulty levels

Level of Difficulty: 1-star

Types of Vulnerabilities Covered:

1. XSS
2. Sensitive Data Exposure
3. Security Misconfiguration
4. Improper Input Validation
5. Unvalidated Redirects
6. Miscellaneous

6 Levels - 100 Challenges



Overall list of vulnerabilities covered in OWASP Juice Shop v14.3.1

1. Broken Access Control - 10 Challenges
2. Broken Anti-Automation - 4 Challenges
3. Broken Authentication - 9 Challenges
4. Cryptographic Issues - 5 Challenges
5. Improper Input Validation - 9 Challenges
6. Injection - 11 Challenges
7. Insecure Deserialization - 2 Challenges
8. Miscellaneous - 5 Challenges
9. Security Misconfiguration - 4 Challenges
10. Security through Obscurity - 3 Challenges
11. Sensitive Data Exposure - 16 Challenges
12. Unvalidated Redirects - 2 Challenges
13. Vulnerable Components - 9 Challenges
14. XSS - 9 Challenges
15. XXE - 2 Challenges

XSS(Cross Site Scripting)

1. DOM XSS
2. Bonus Payload

Sensitive Data Exposure

1. Confidential Document
2. Exposed Metrics

Security Misconfiguration

1. Error handling

Improper Input Validation

1. Missing Encoding
2. Repetitive Registration

Unvalidated Redirects

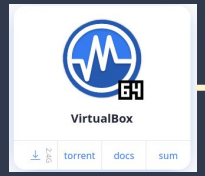
1. Outdated Allowlist

Miscellaneous

1. Bully Chatbot
2. Mass Dispel
3. Privacy Policy
4. Score-board

OWASP Juice Shop (v14.3.0); Difficulty Level: 1-star (⭐) - 6 Types of Vulnerabilities Covered

Setting up OWASP Juice Shop (v14.5.1) From Sources

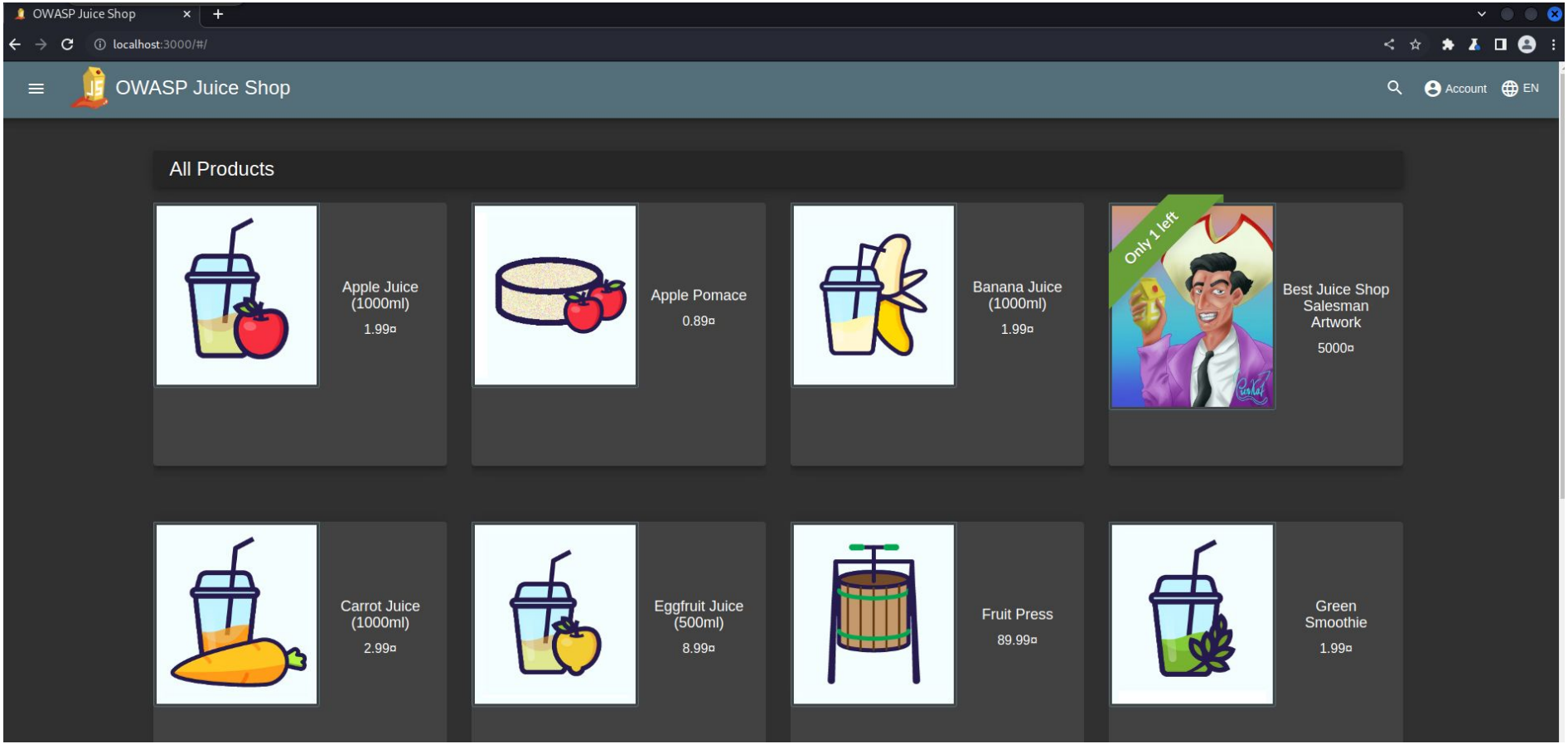



```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/OWASPJuiceShop]
└─$ cd juice-shop

┌──(kali㉿kali)-[~/Desktop/OWASPJuiceShop/juice-shop]
└─$ npm start

> juice-shop@14.3.0 start
> node build/app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated (OK)
info: Detected Node.js version v18.7.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Entity models 19 of 19 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file polyfills.js is present (OK)
info: Required file tutorial.js is present (OK)
info: Required file main.js is present (OK)
info: Required file vendor.js is present (OK)
info: Required file runtime.js is present (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000
info: Restored 1-star directoryListingChallenge (Confidential Document)
info: Restored 1-star localXssChallenge (DOM XSS)
info: Restored 1-star errorHandlingChallenge (Error Handling)
info: Restored 1-star redirectCryptoCurrencyChallenge (Outdated Allowlist)
info: Restored 1-star privacyPolicyChallenge (Privacy Policy)
info: Restored 1-star passwordRepeatChallenge (Repetitive Registration)
info: Restored 1-star scoreBoardChallenge (Score Board)
info: Restored 1-star missingEncodingChallenge (Missing Encoding)
info: Restored 1-star exposedMetricsChallenge (Exposed Metrics)
info: Restored 1-star xssBonusChallenge (Bonus Payload)
info: Restored 1-star bullyChatbotChallenge (Bully Chatbot)
info: Restored 1-star closeNotificationsChallenge (Mass Dispel)
^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A
```

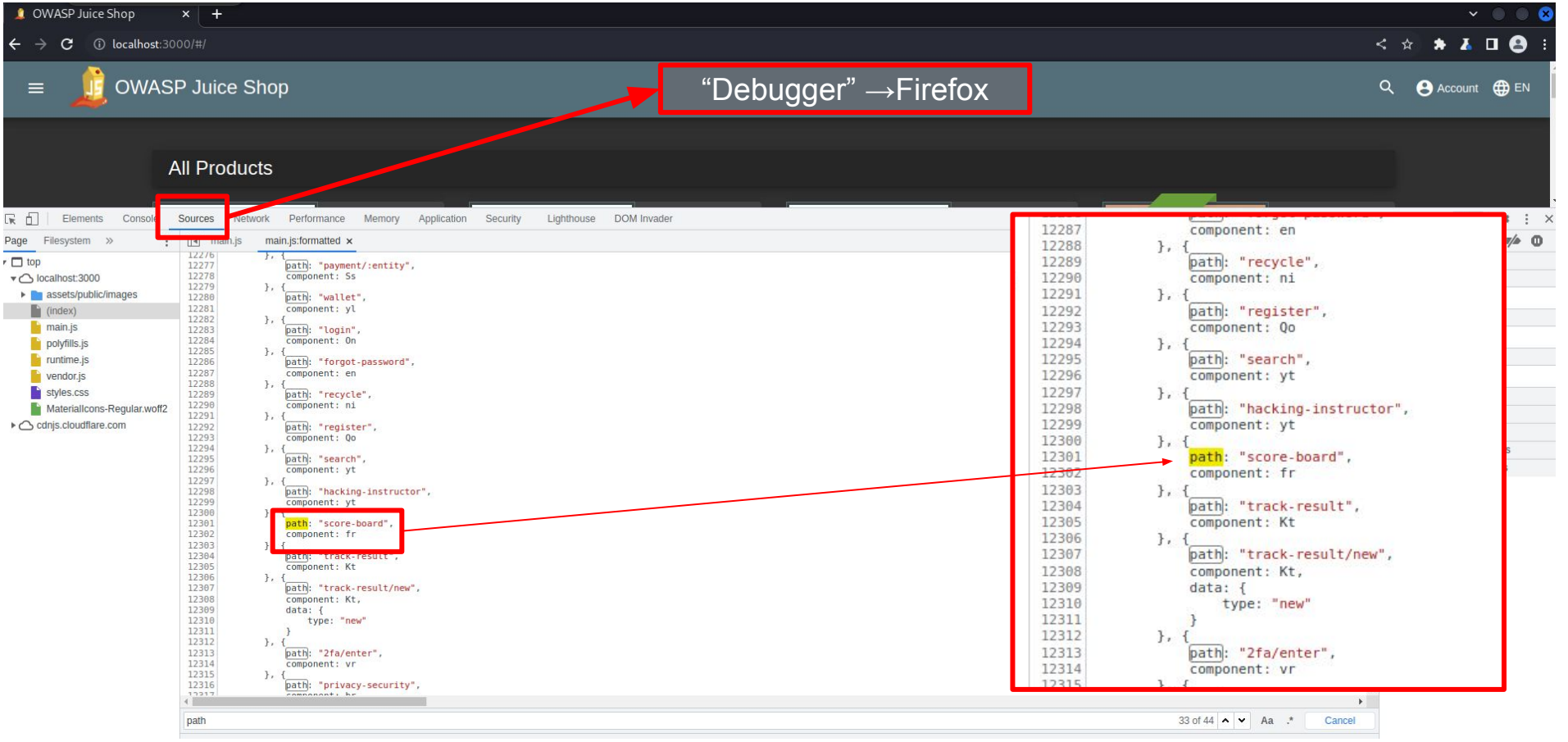
https://github.com/juice-shop/juice-shop/tree/v14.3.1#setup

Steps for setting up OWASP Juice Shop on Kali Linux VM

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



http://localhost:3000

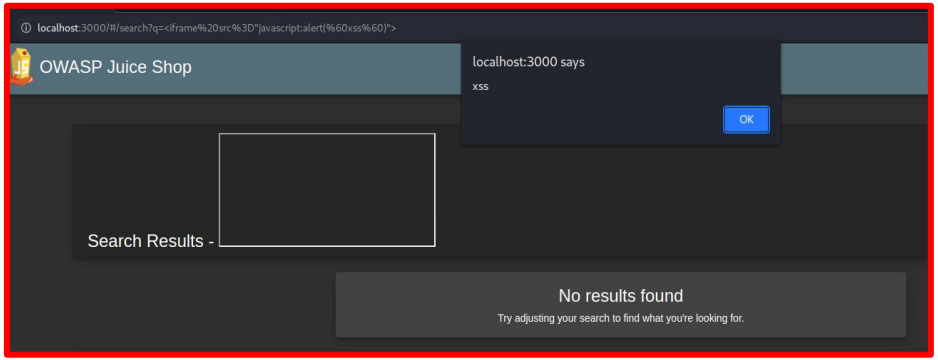


$ Hint : Getting the "*score-board*"

XSS(Cross Site Scripting)

1. DOM XSS
2. Bonus Payload

→ DOM(Document Object Model) based XSS



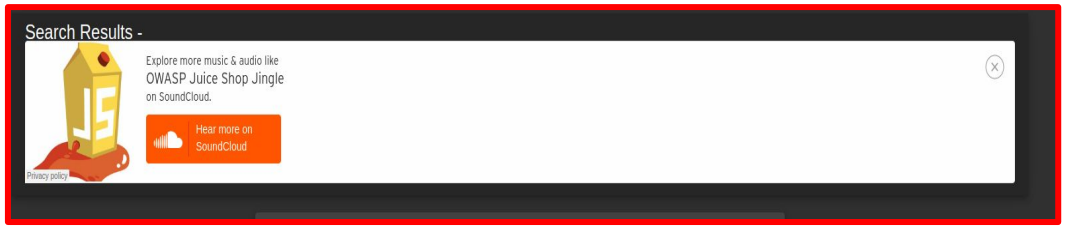

```
<iframe src="javascript:alert(`xss`)">
```

XSS: #Challenge: 1. *DOM XSS*

XSS(Cross Site Scripting)

1. DOM XSS
2. Bonus Payload

DOM(Document Object Model) based XSS

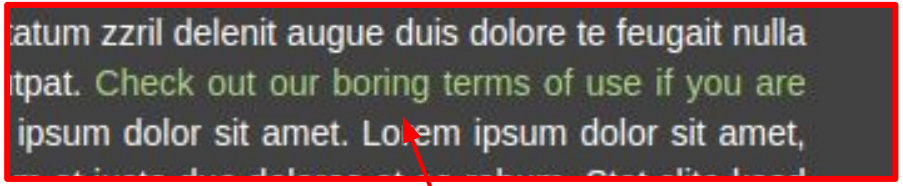


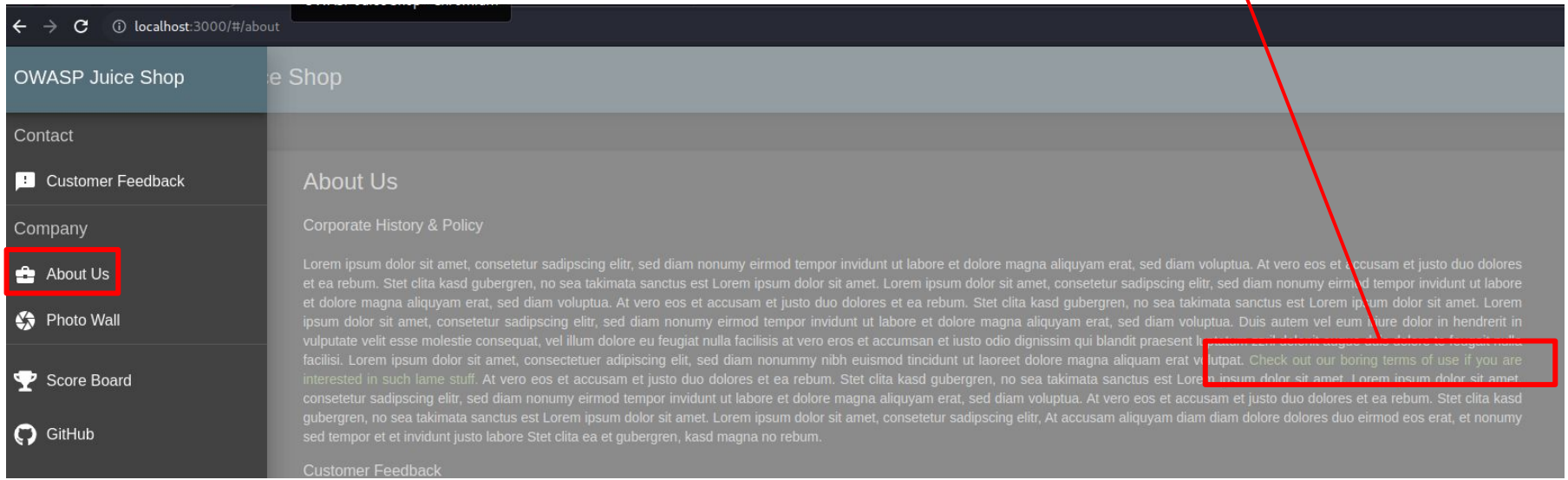

```
<iframe width="100%" height="166"
scrolling="no" frameborder="no"
allow="autoplay"
src="https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true"></iframe>
```

XSS: #Challenge: 2. *Bonus Payload*

Sensitive Data Exposure

1. Confidential Document
2. Exposed Metrics

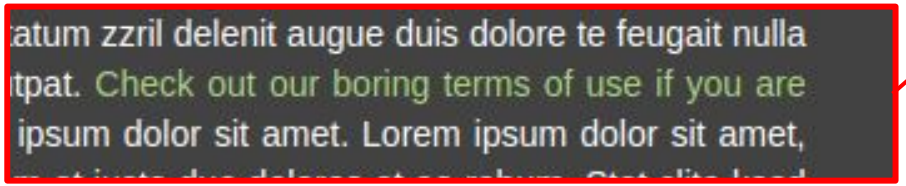


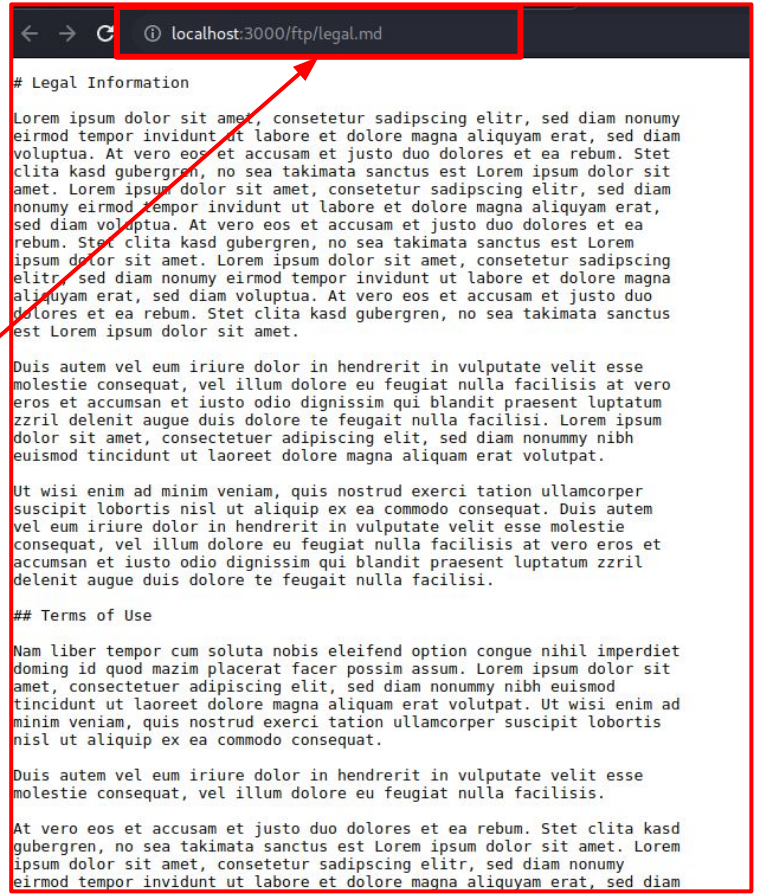


Sensitive Data Exposure: #Challenge: 1. *Confidential Document*

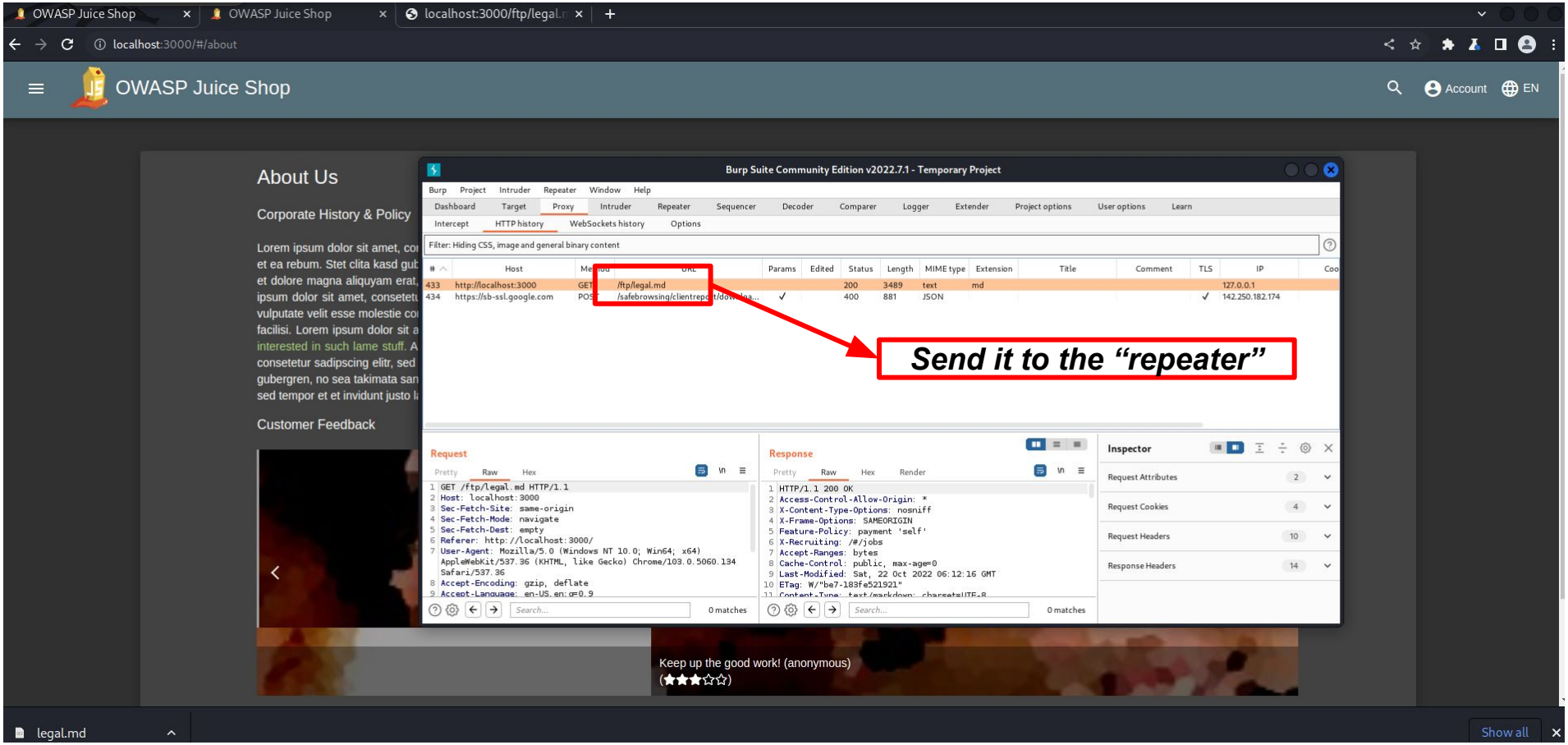
Sensitive Data Exposure

1. Confidential Document
2. Exposed Metrics

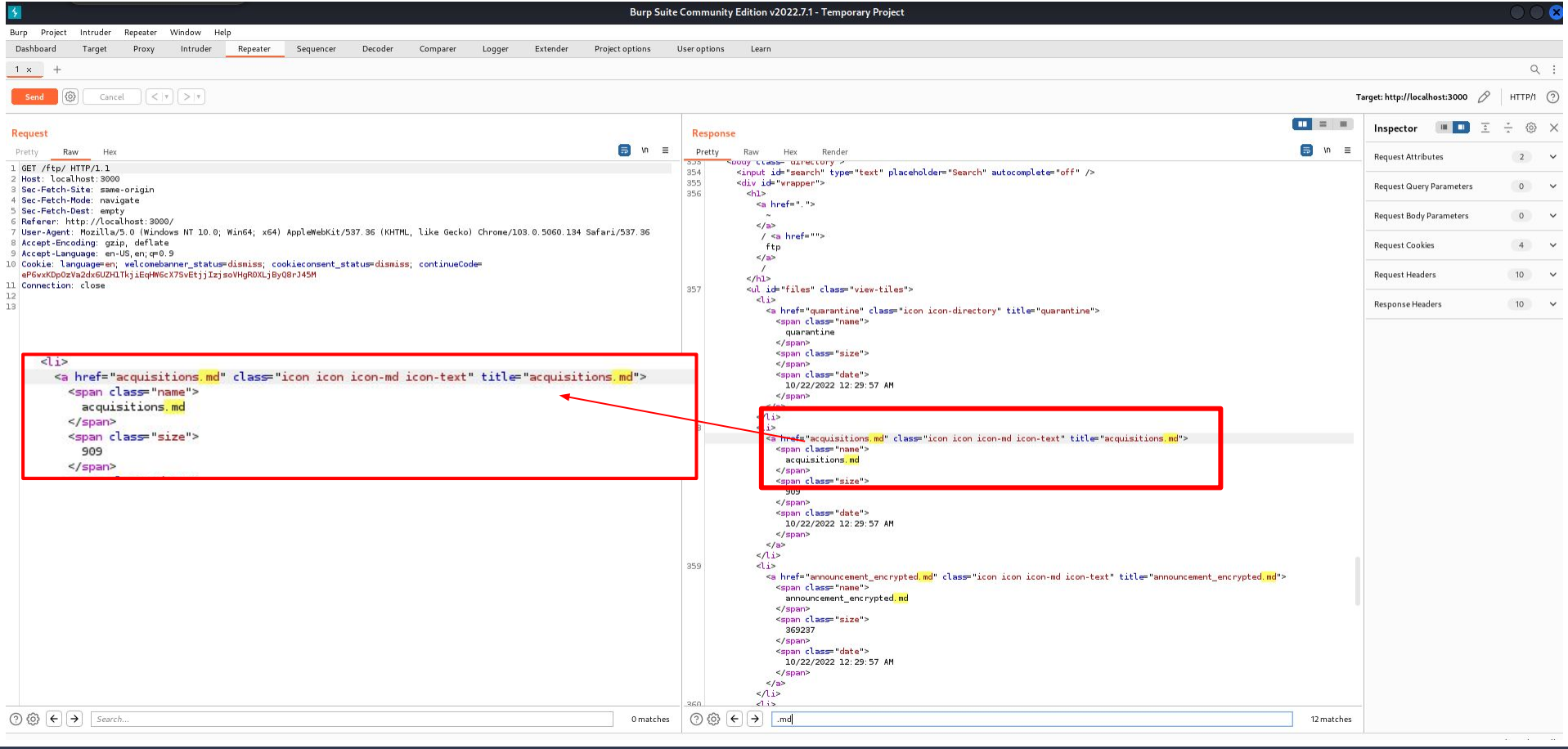




Sensitive Data Exposure: #Challenge: 1. *Confidential Document*



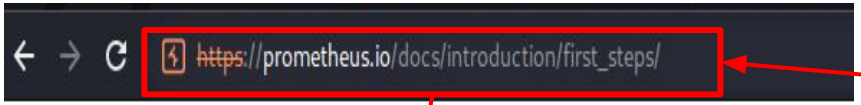
Sensitive Data Exposure: #Challenge: 1. *Confidential Document*



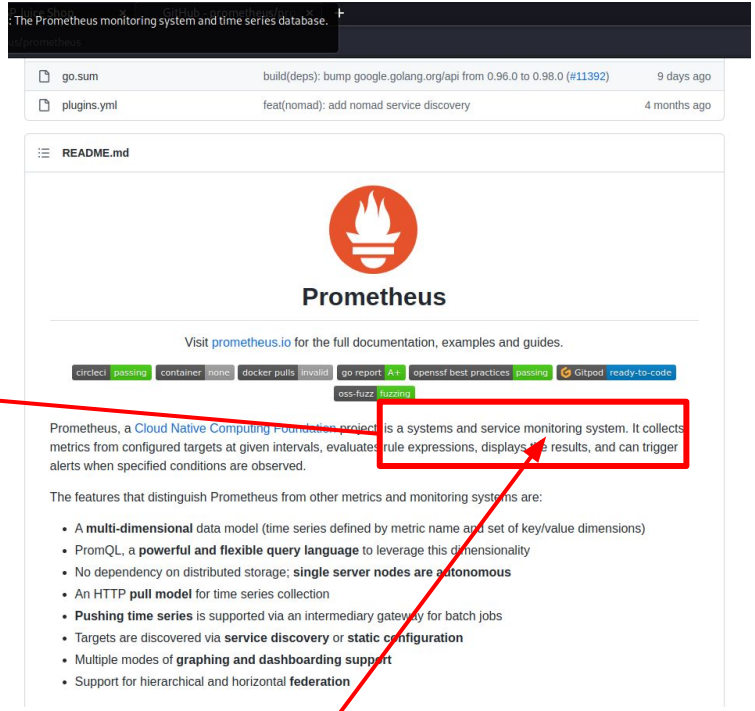
Sensitive Data Exposure: #Challenge: 1. *Confidential Document*

Sensitive Data Exposure

1. Confidential Document
2. Exposed Metrics

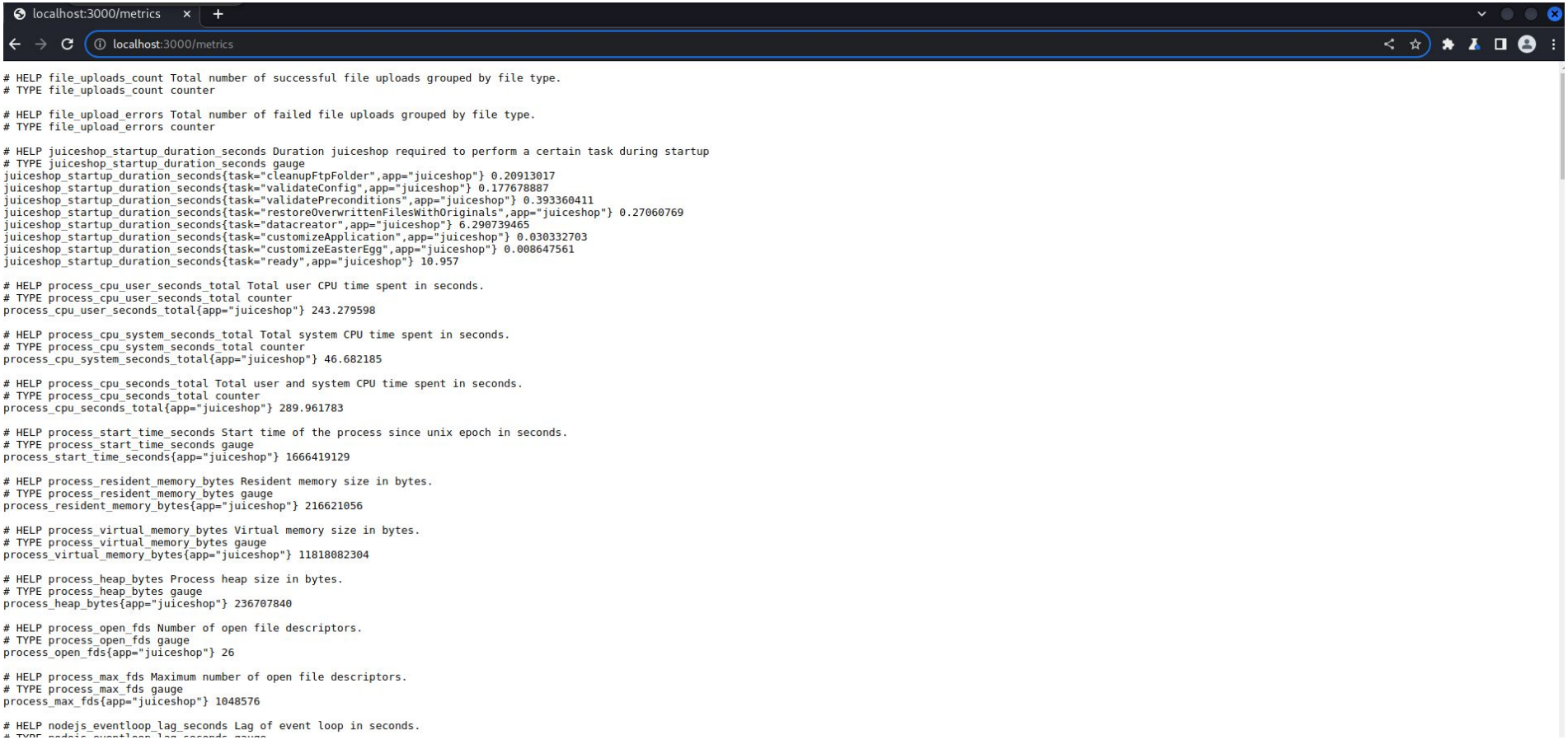


The last block, `scrape_configs`, controls what resources Prometheus monitors. Since Prometheus also exposes data about itself as an HTTP endpoint it can scrape and monitor its own health. In the default configuration there is a single job, called `prometheus`, which scrapes the time series data exposed by the Prometheus server. The job contains a single, statically configured, target, the `localhost` on port `9090`. Prometheus expects metrics to be available on targets on a path of `/metrics`. So this default job is scraping via the URL: http://localhost:9090/metrics.



Exposed Metrics	★	Find the endpoint that serves usage data to be scraped by a popular monitoring system.	Sensitive Data Exposure
Mass Dispel	★	Close multiple "Challenge solved"-notifications in one go.	Miscellaneous

Sensitive Data Exposure: #Challenge: 2. *Exposed Metrics*

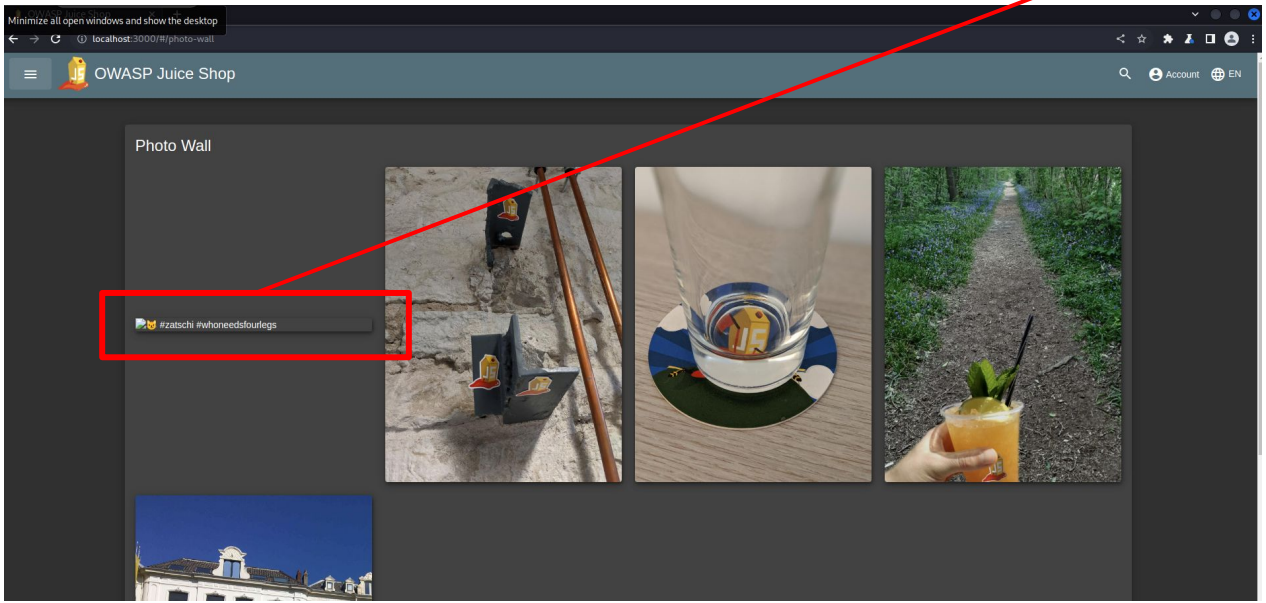


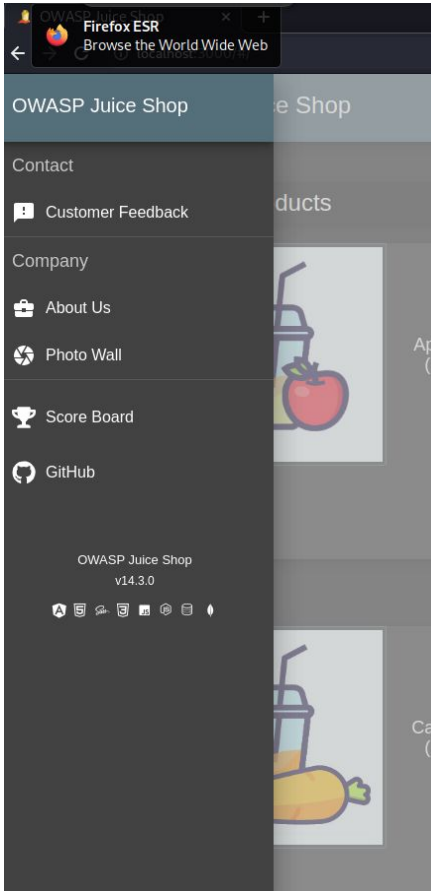
Sensitive Data Exposure: #Challenge: 2. *Exposed Metrics*

Improper Input Validation

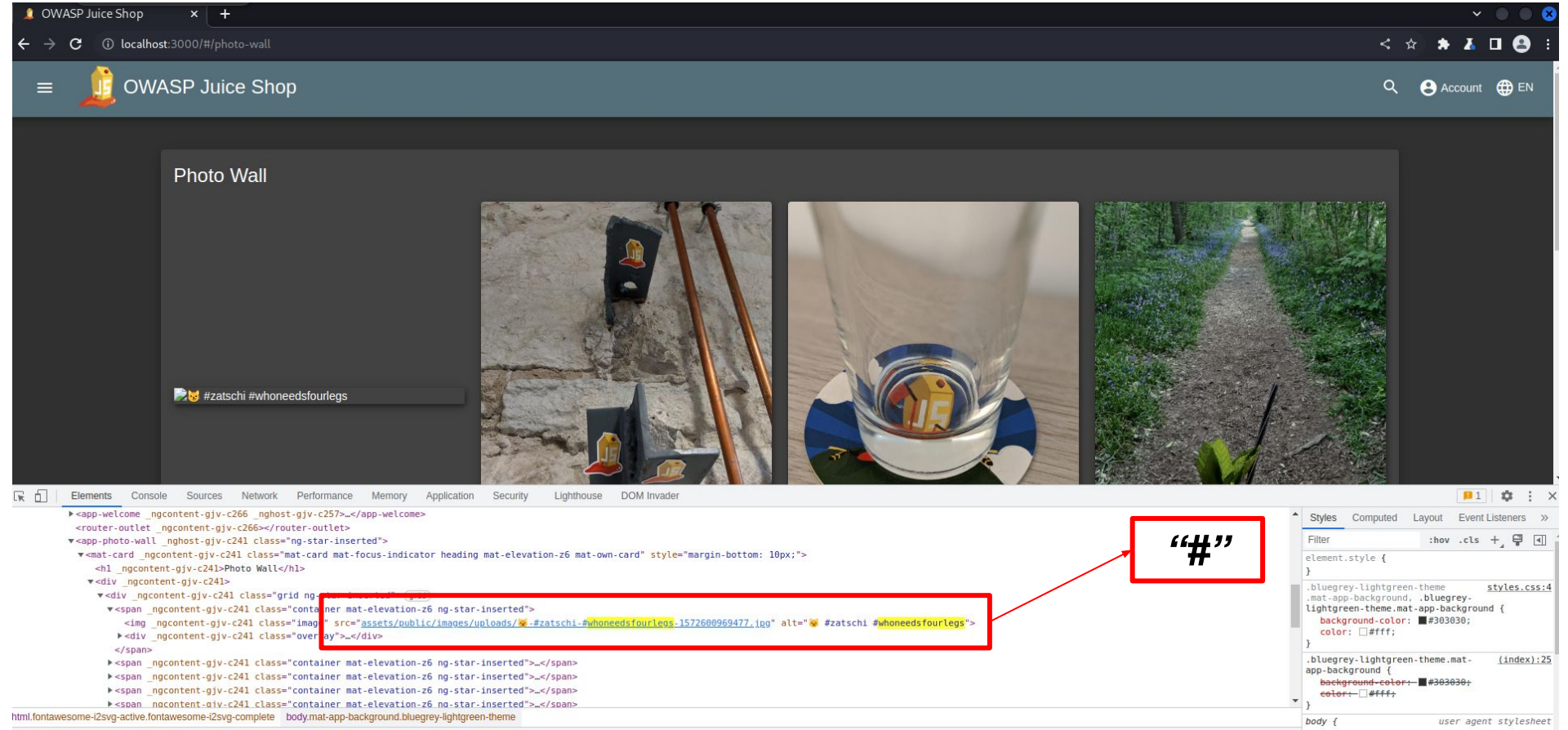
1. Missing Encoding
2. Repetitive Registration

“Inspect Element”

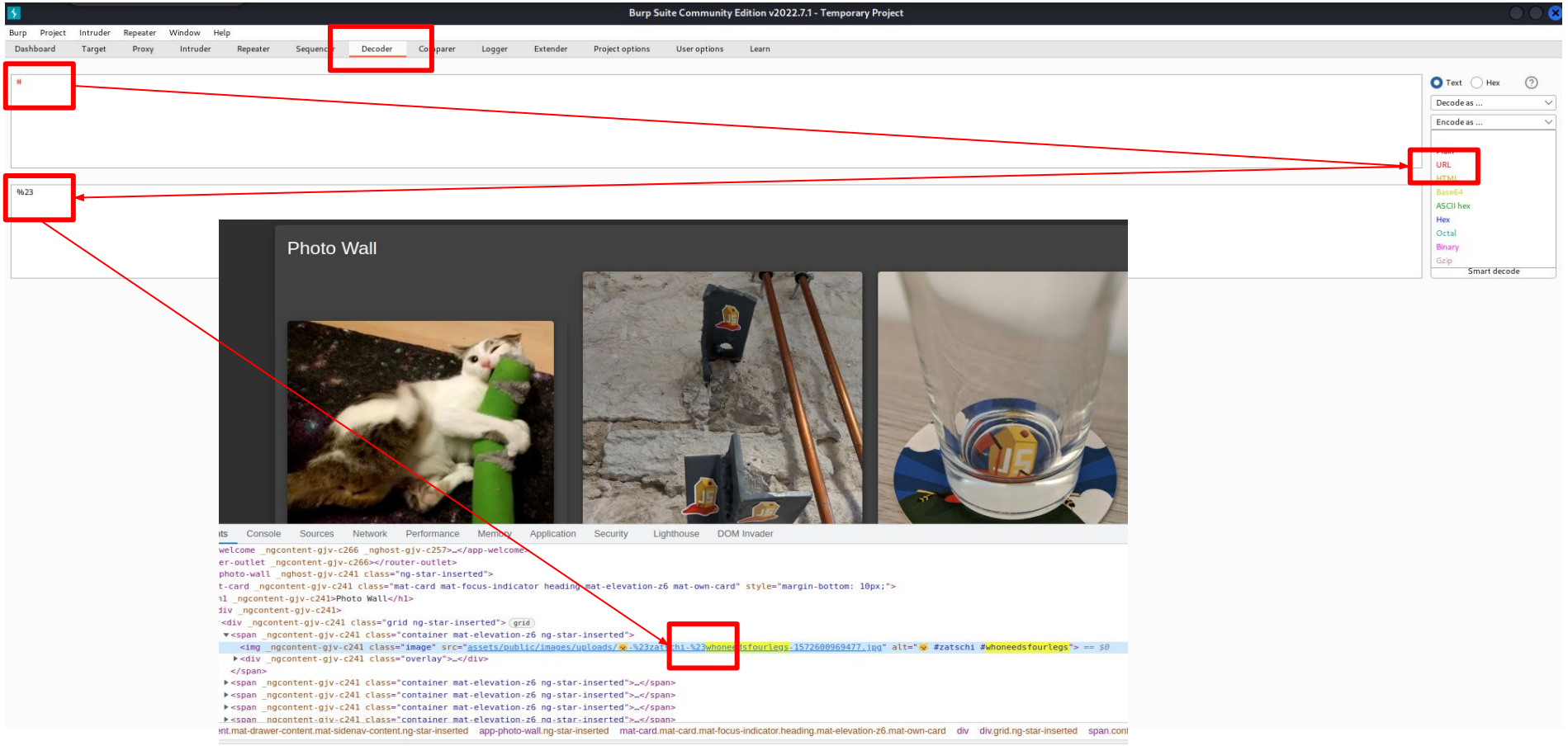




Improper Input Validation: #Challenge 1: *Missing Encoding*



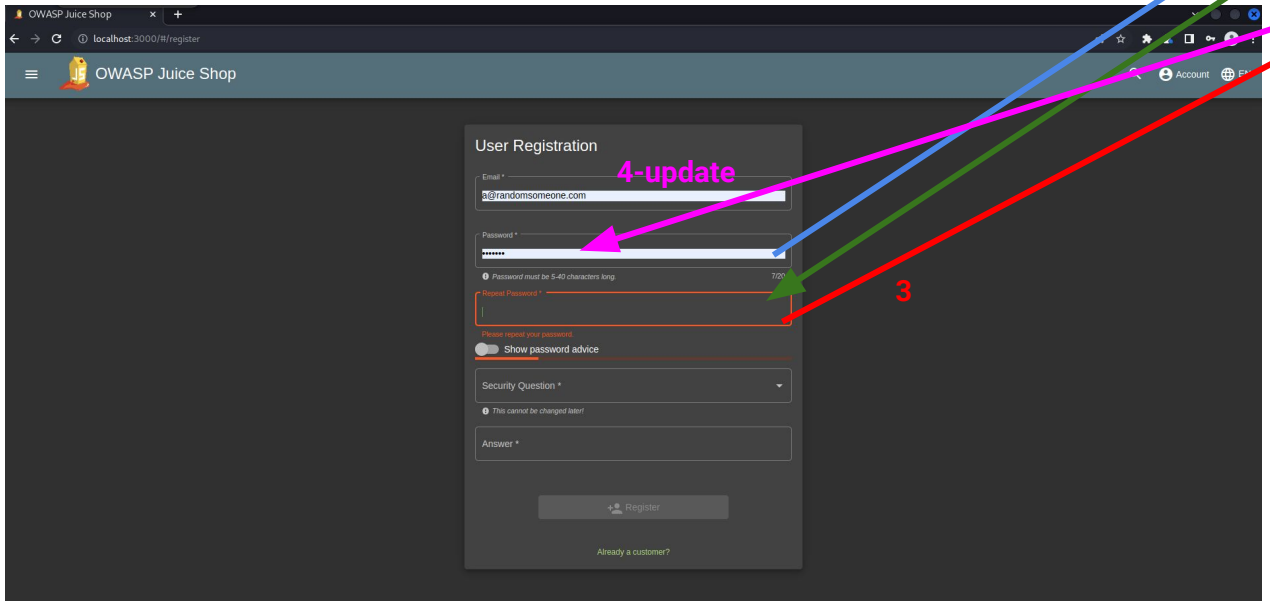
Improper Input Validation: #Challenge 1: *Missing Encoding*



Improper Input Validation: #Challenge 1: *Missing Encoding*

Improper Input Validation

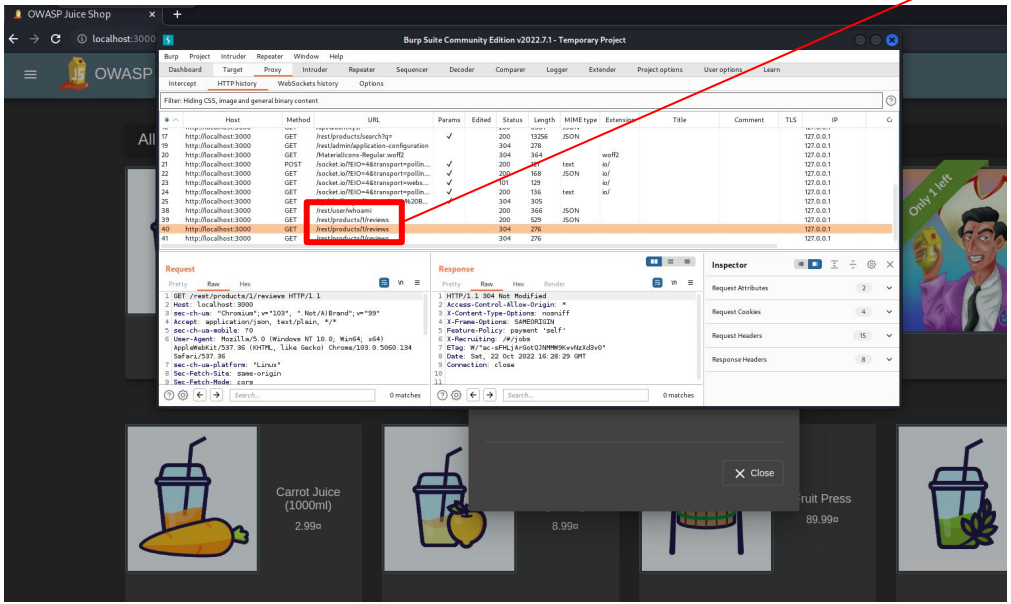
1. Missing Encoding
2. Repetitive Registration

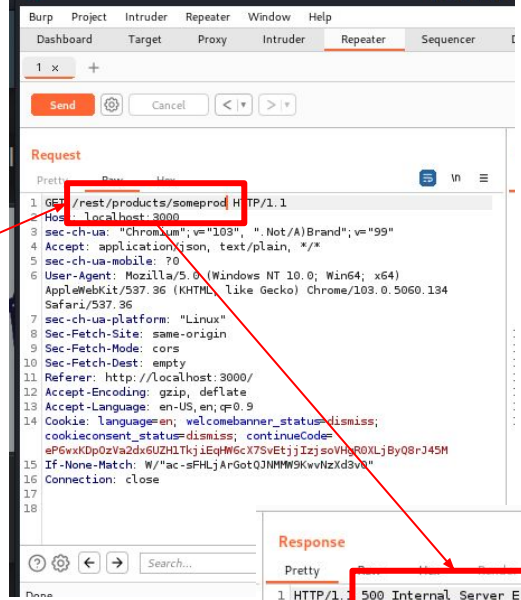


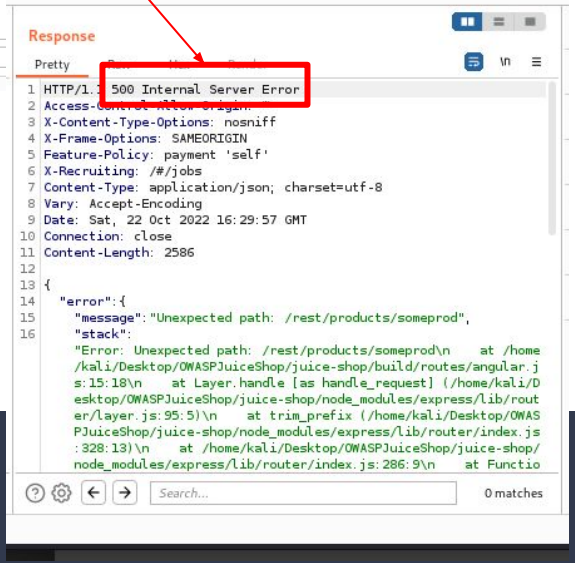
Improper Input Validation: #Challenge 2: *Repetitive Registration*

Security Misconfiguration

1. Error handling



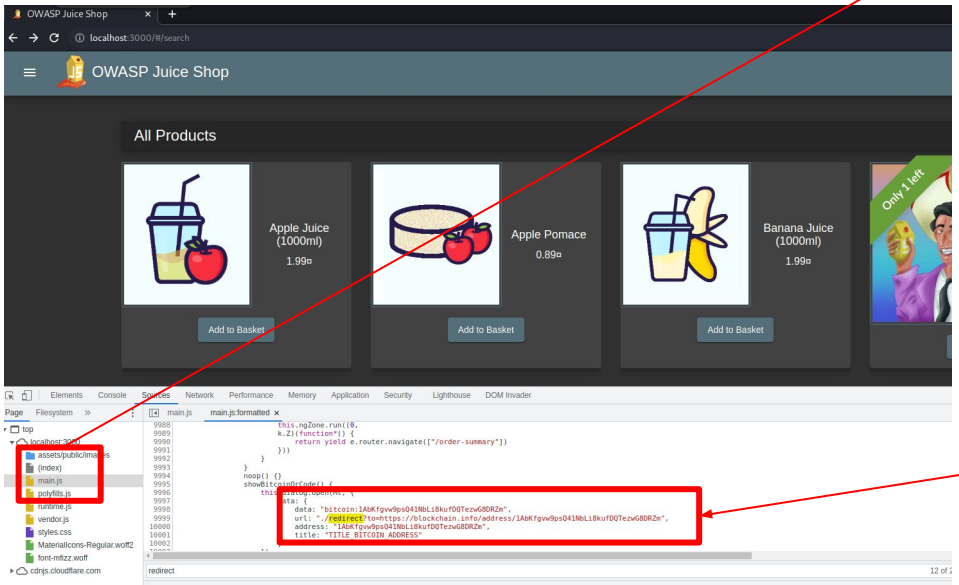


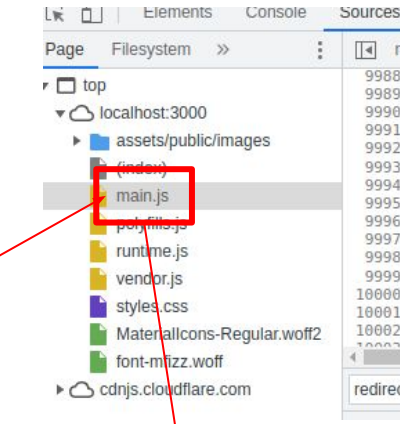


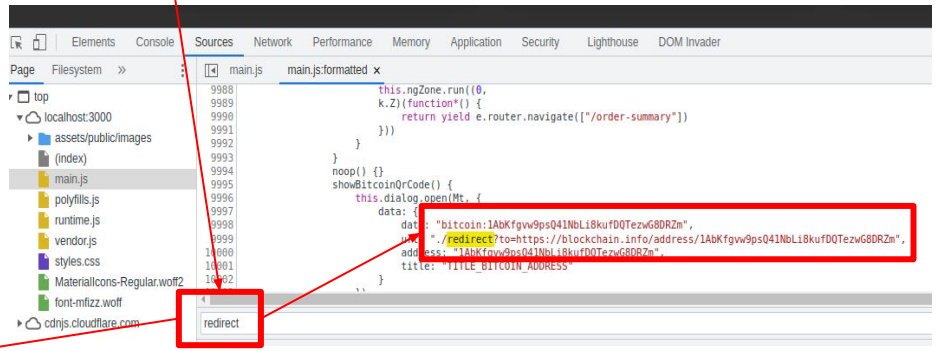
Security Misconfiguration: #Challenge - Error Handling

Unvalidated Redirects

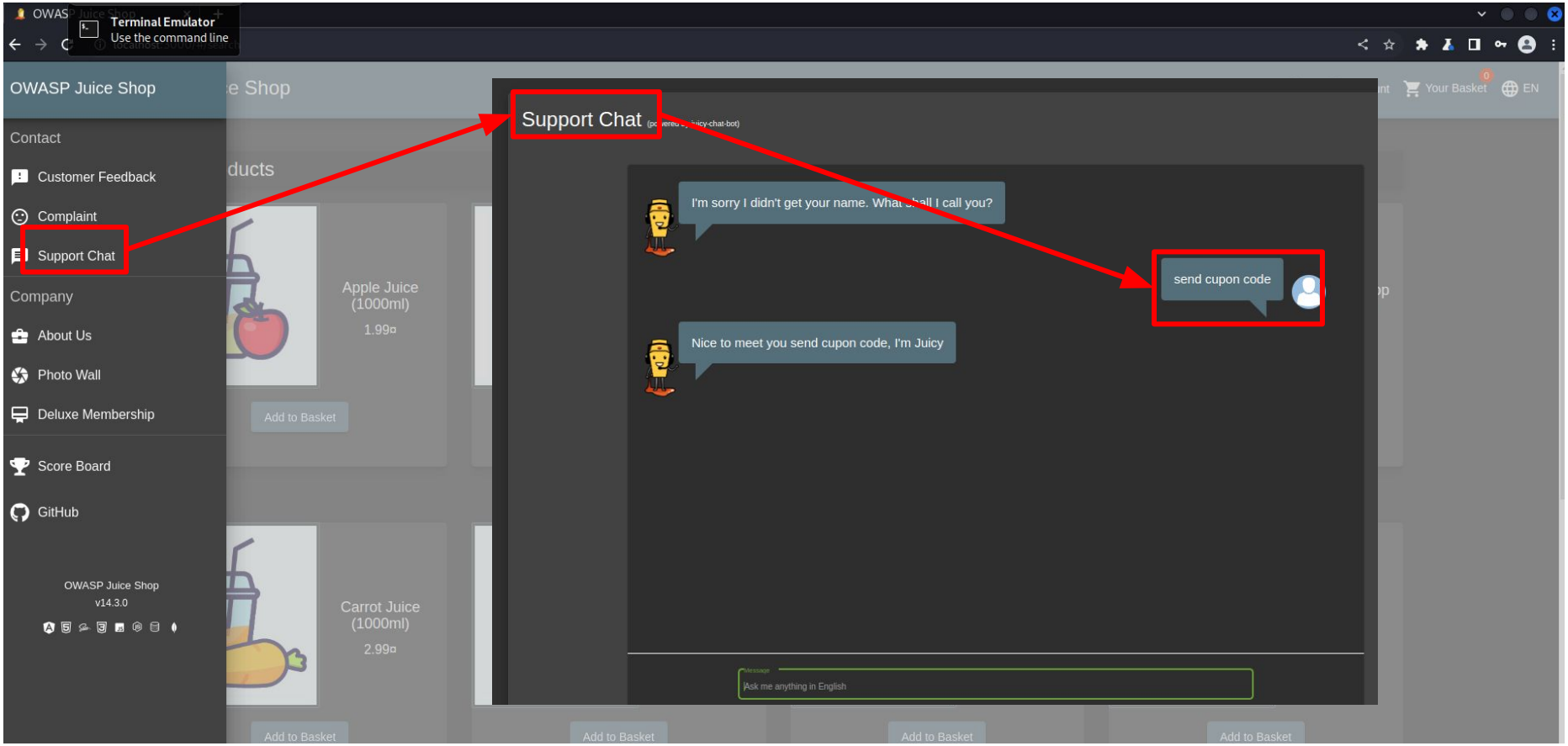
1. Outdated Allowlist



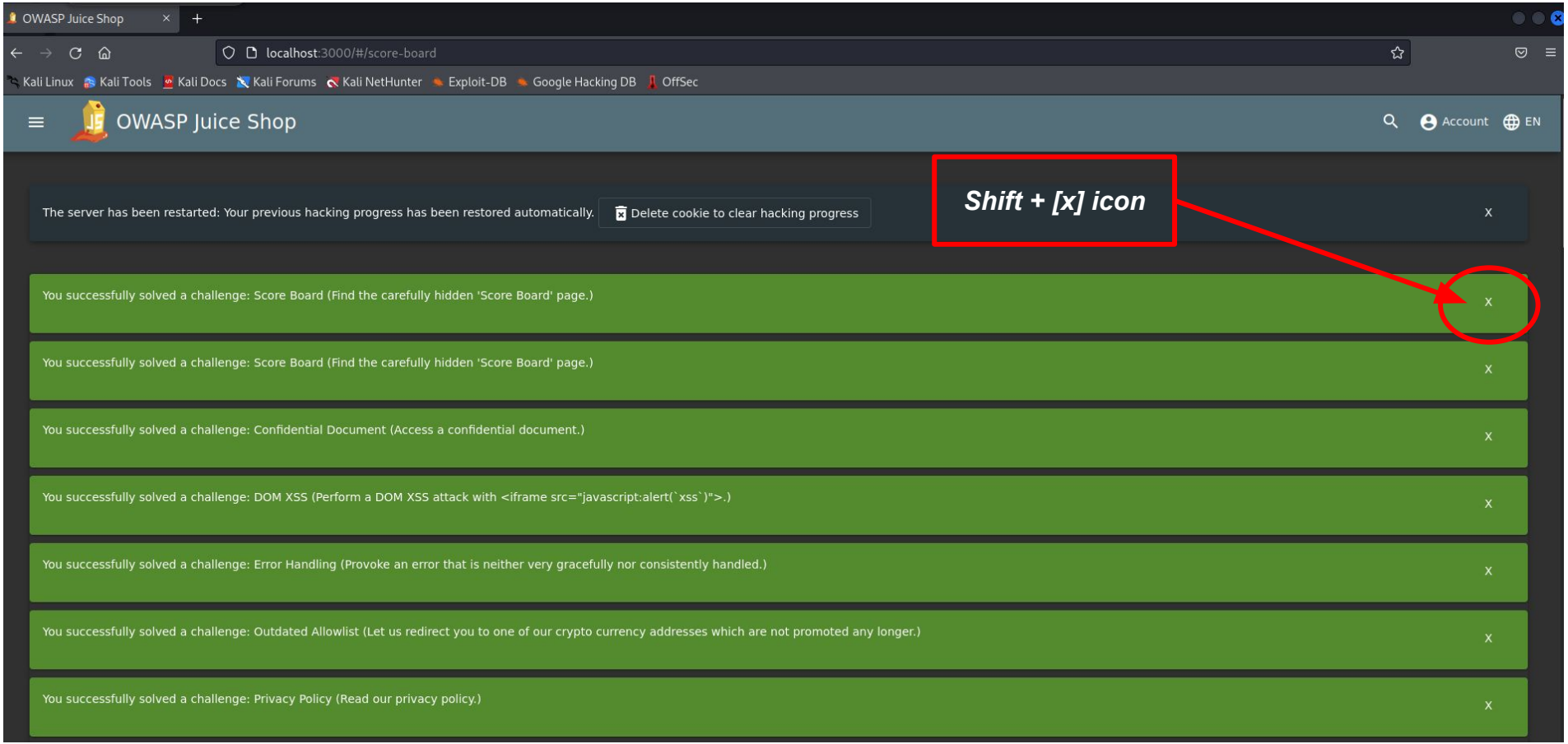




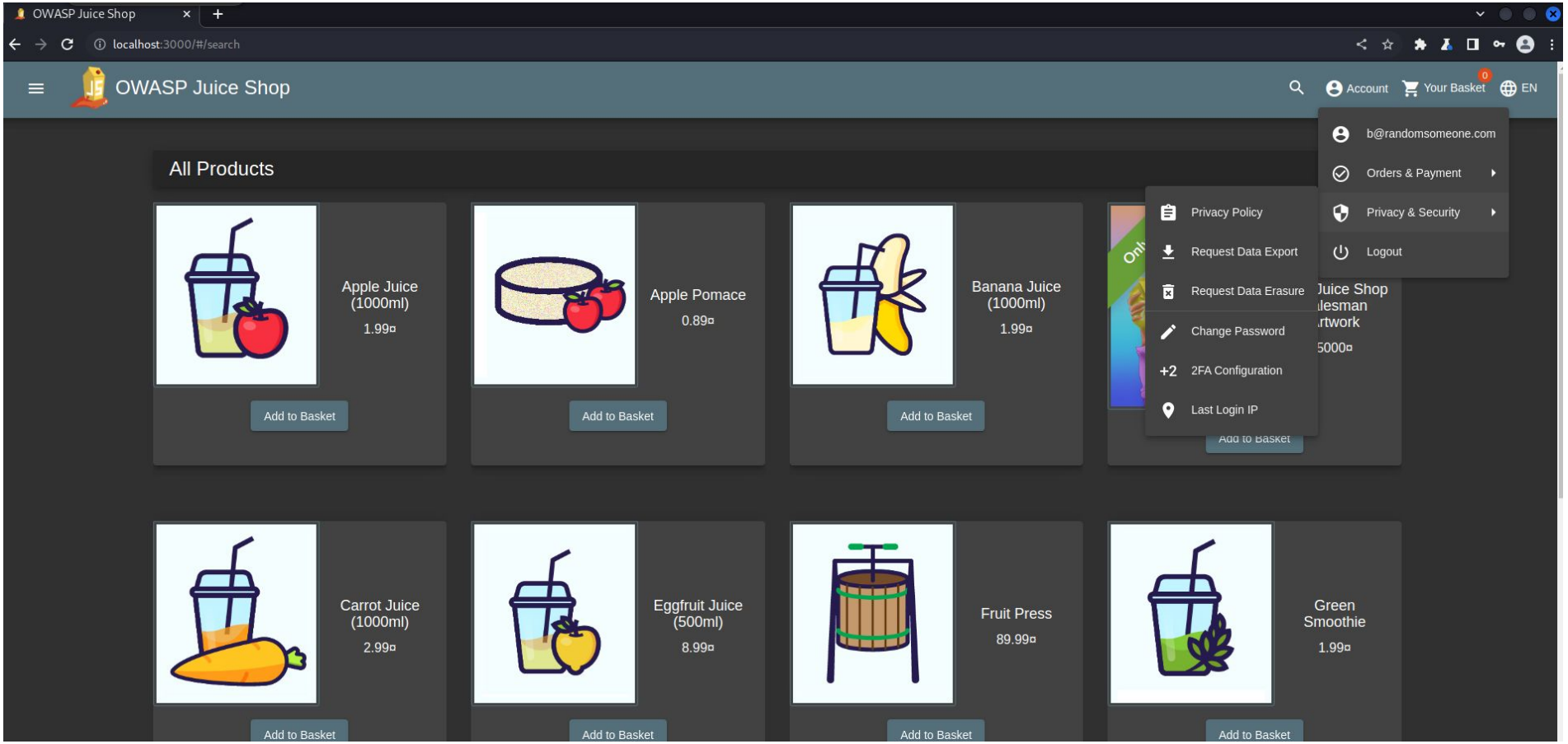
Unvalidated Redirects: #Challenge - 1. *Outdated Whitelist*



Miscellaneous: #Challenge - 1. *Bully Chatbot*



Miscellaneous: #Challenge - 2. *Mass Dispel*



Miscellaneous: #Challenge - 3. *Privacy Policy*

OWASP Juice Shop v14.3.1

Coding Challenge

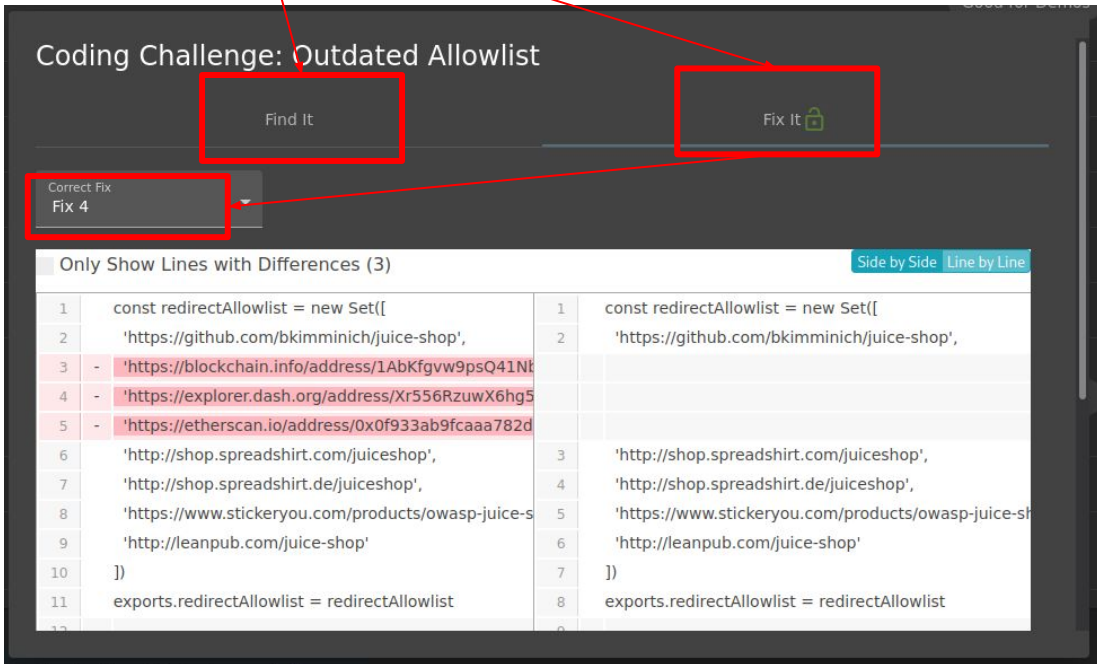
Difficulty Level: 1-star

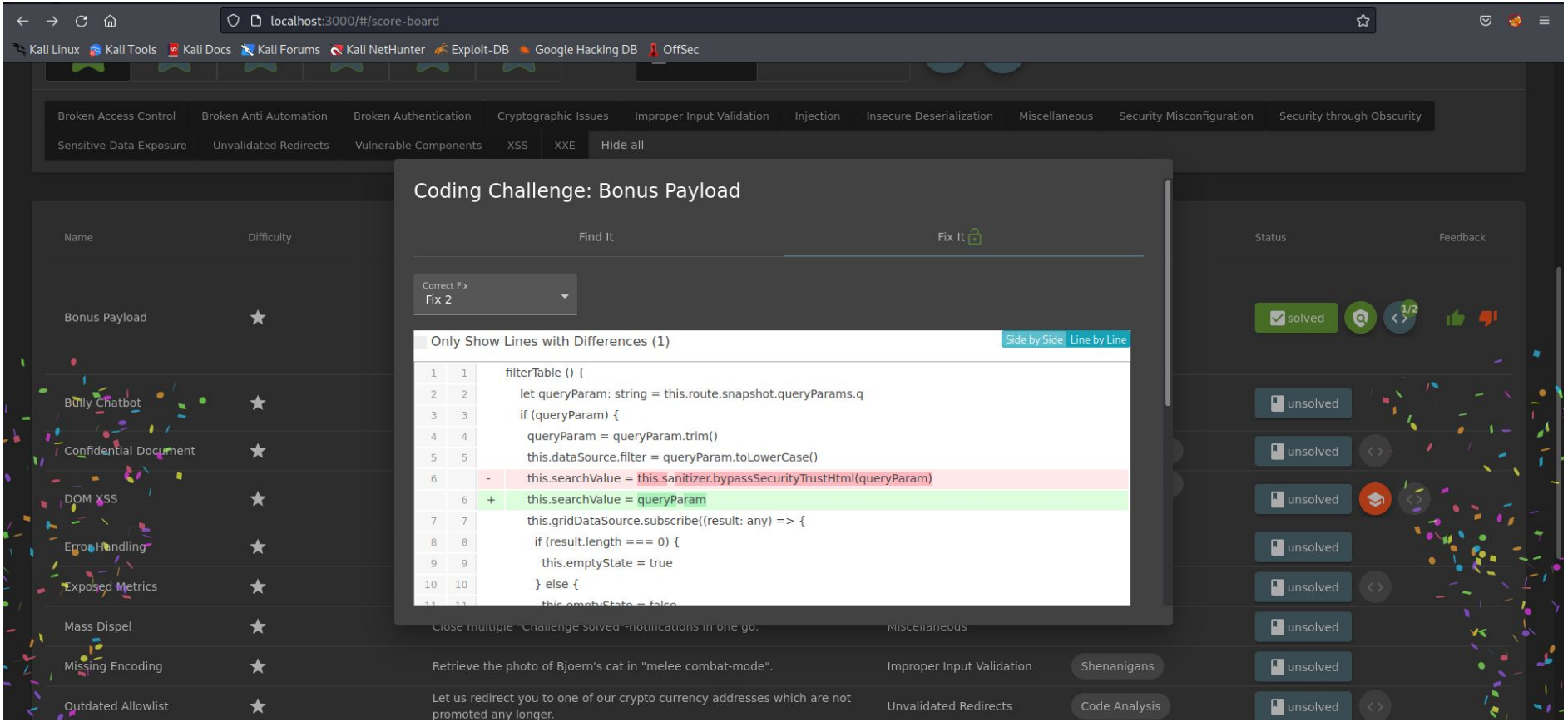
"6 Coding Challenges"

Two phases of coding challenge:

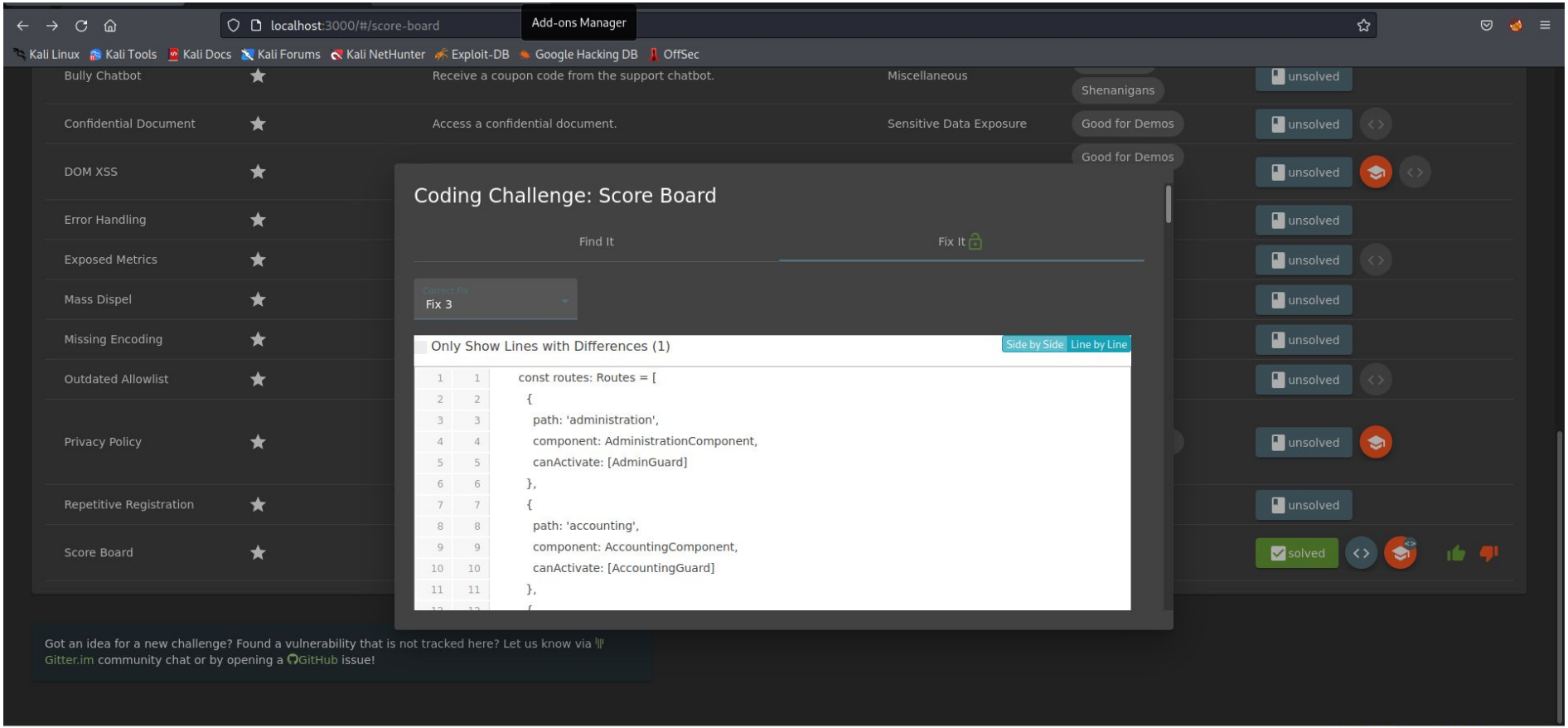
I. "Find it" - *Select the vulnerable line(s) of code*

II. "Fix it" - *Select the correct solution for the vulnerable code that you have selected.*

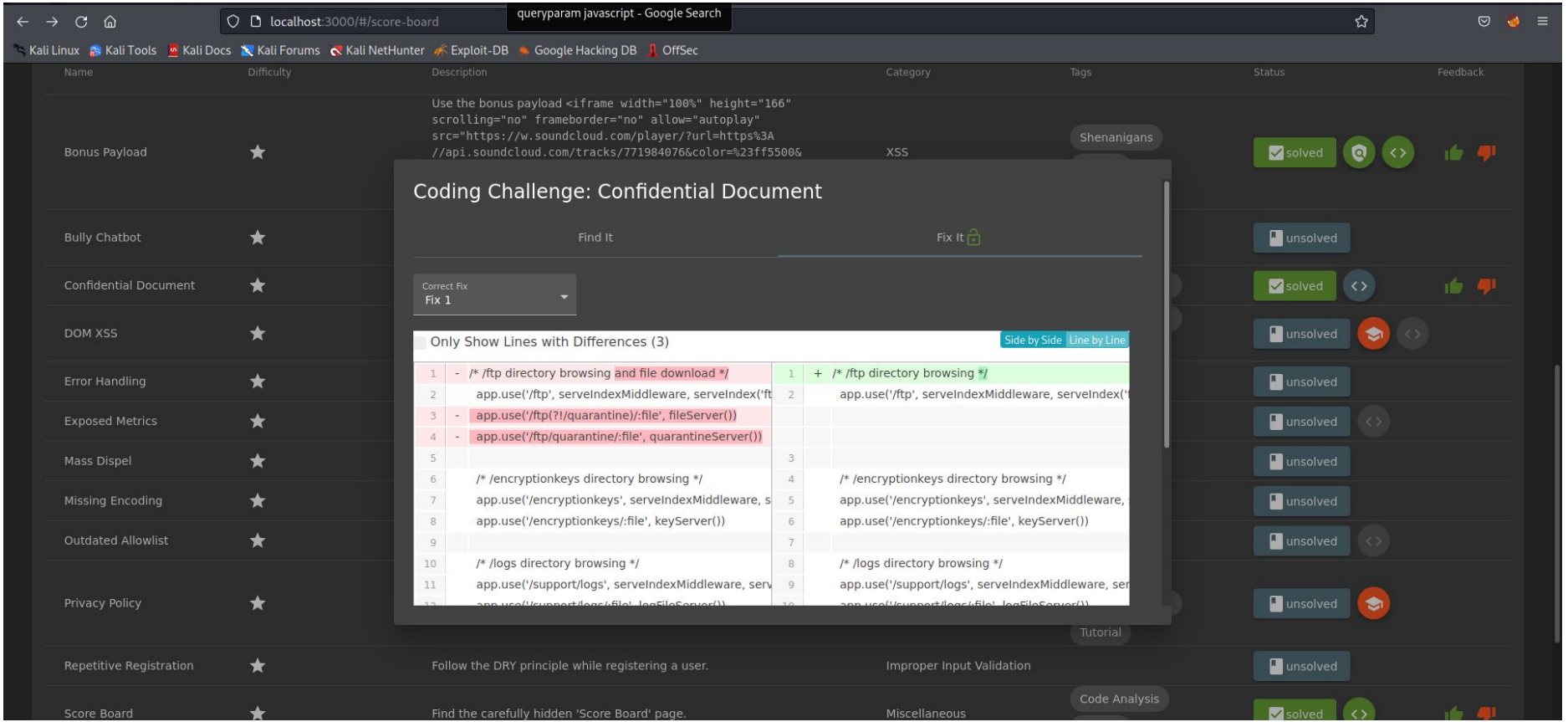




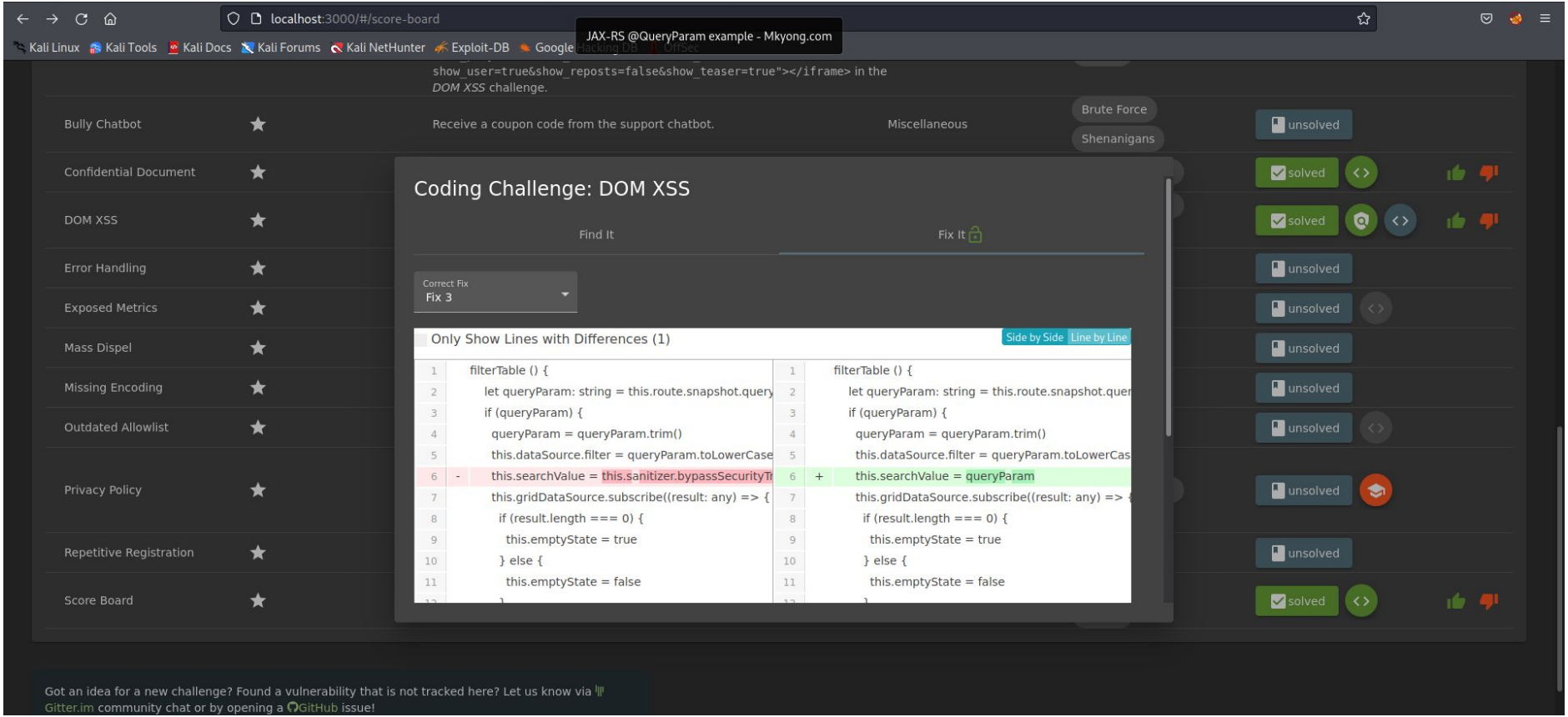
Coding Challenge #1: *Bonus Payload*



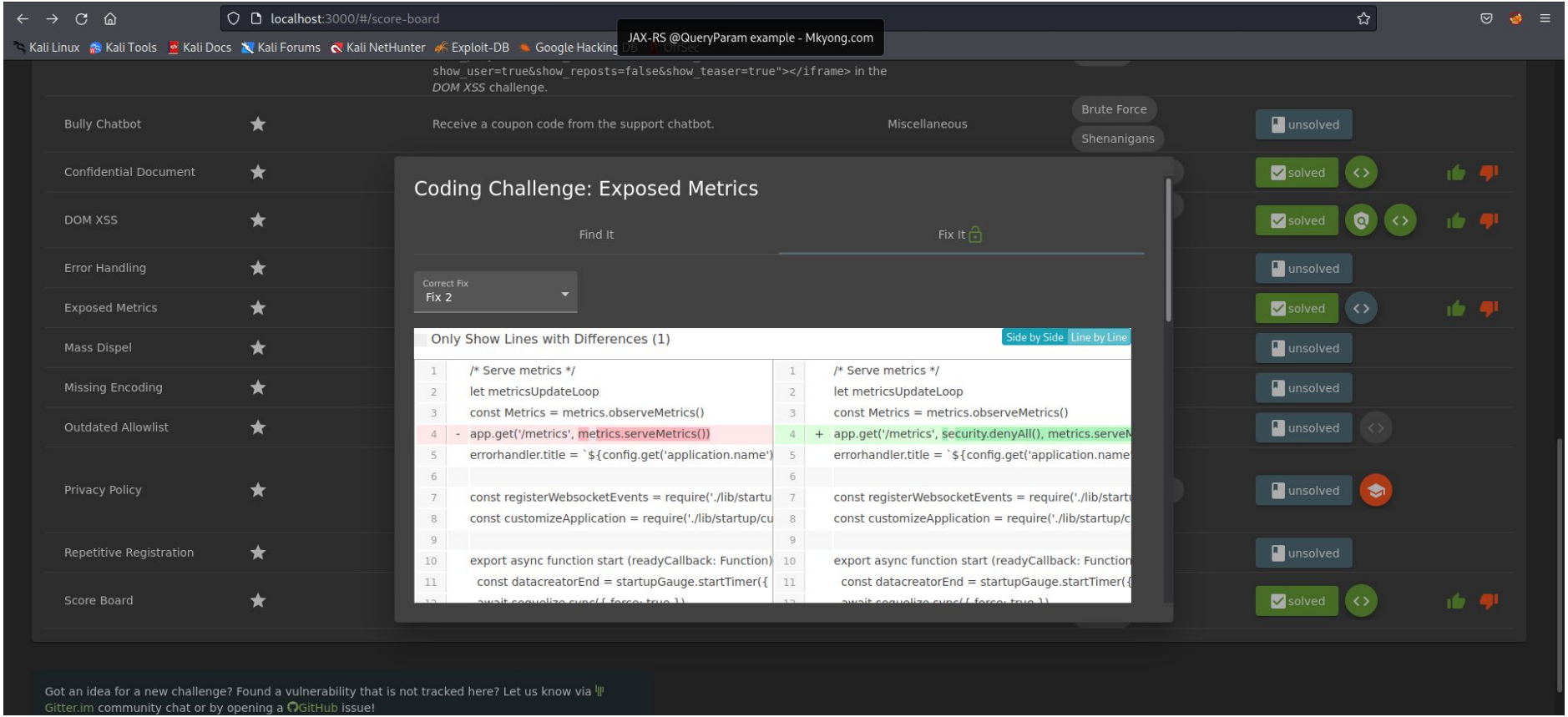
Coding Challenge #2: *Score Board*



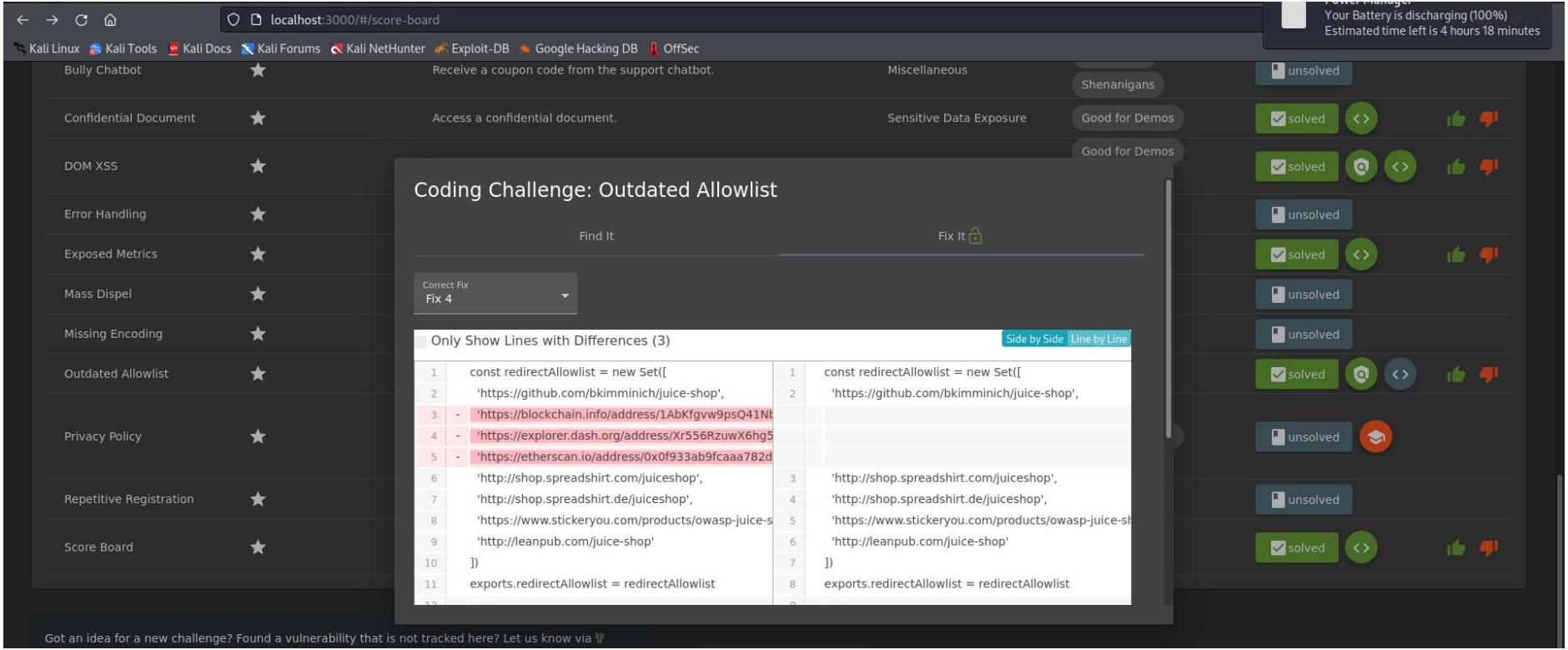
Coding Challenge #3: *Confidential Document*



Coding Challenge #4: *DOM XSS* (*Similar to "Bonus Payload Challenge"*)



Coding Challenge #5: *Exposed Metrics*



Coding Challenge #6: *Outdated Allowlist*