

Redo today's hack, **on your own.**

Everything from Tuesday in one place: get the lab running, understand the two bugs, learn to hunt them yourself, and practise before Thursday. **Work through it tonight and you'll walk into the challenge ready.**

Mohamed Medjahdi, Cybersecurity Engineer • medjahdi.dev

CONTENTS

- | | |
|---------------------------------|----------------------------|
| 01 Set up your lab | 02 XSS, with the code |
| 03 SQL injection, with the code | 04 How to hunt on your own |
| 05 Practice tonight | 06 The report template |

Set up your lab

Three pieces: **Docker** runs the target, **Juice Shop** is the target, **Burp** lets you read and change requests. Do this once and it's ready forever.

Install Docker, then check it

Get Docker Desktop (Mac and Windows), open it once, wait for the whale icon, then:

```
terminal
```

```
docker --version    # any version is fine
```

Run the target: OWASP Juice Shop

```
terminal, then open localhost:3000
```

```
docker run --rm -p 3000:3000 bkimminich/juice-shop  
# open http://localhost:3000
```

Open Burp Suite

Install Burp Suite Community (free). Use its built-in browser, and browse to the lab through **localtest.me** so the traffic flows cleanly through Burp:

```
in Burp's browser, go to
```

```
http://localtest.me:3000
```

If you're stuck

- › **Docker command fails:** open Docker Desktop first and wait for it to finish starting.
- › **Port 3000 already in use:** run it on another port, **-p 3001:3000**, then open localhost:3001.
- › **Burp sees no traffic:** use Burp's own browser (Proxy tab, Open Browser), not your normal one.
- › **Still stuck:** paste the error into an AI and ask what it means, then verify. Nothing here can break your computer.

XSS, with the code

XSS happens when a site drops your input **into the page without cleaning it**, so the browser runs it as code.

the vulnerable code, the bug is here

```
html = "<p>Results for: " + userInput + "</p>"
# normal:  userInput = "shoes"           → fine
# attack:  userInput = "<script>...</script>" → runs
```

the payload we used, paste into the search bar

```
<iframe src="javascript:alert(document.cookie)">
```

Why it matters: document.cookie is the victim's login session. Stealing it means logging in as them.

The fix (blue team): encode output, so < is shown as text and never runs.

SQL injection, with the code

Same bug, different place. To log you in, the site builds a database question by **gluing your input into text**. Change the input, change the question.

```
the vulnerable code, the bug is here
```

```
query = "...WHERE email='" + input + "' AND pass='"...  
# you type: input = ' OR 1=1--  
# becomes: WHERE email='' OR 1=1 -- (password ignored, always true)
```

```
the payload we used, email field, any password
```

```
' OR 1=1--
```

If it doesn't work, try these variants, databases differ: ' OR 1=1;-- or admin@juice-sh.op'--

The fix (blue team): use parameterized queries, input is sent as data, never as part of the command.

How to hunt on your own

This is the real skill, and what Thursday's challenge needs. For **any** input on a site you're allowed to test, ask:

1 **Does my text show up on the page?**

Type `<b>test</b>`. If it turns bold, the site isn't cleaning input, try XSS.

2 **Does it go to a database?**

Any login or search. Type a single quote `'`. If you get an error, SQLi is likely.

3 **Watch it in Burp.**

Send the request to Repeater, change one thing, resend, read the response. Repeat.

4 **Think like the restaurant.**

What is this input allowed to do, and what happens if I ask for more?

5 **When stuck.**

Ask AI "why didn't this payload work here?", then verify what it says yourself.

The goal isn't memorizing payloads. It's the habit: **poke every input and watch what the server does.**

Practice tonight (free)

The best way to lock this in before Thursday. Same two vulnerabilities, guided, in your browser:

- › **PortSwigger · XSS labs**, start with the reflected XSS apprentice labs.
- › **PortSwigger · SQL injection labs**, start with the login-bypass apprentice lab.
- › **Your Juice Shop**, keep hunting. It has a built-in challenge scoreboard at [/#/score-board](#).

Web Security Academy ↗

Juice Shop ↗

The report template

Finding a bug is half the job. Thursday you'll write one of these. Paste your finding and this template into any AI for a draft, then **check every line yourself**.

```
vulnerability-report.md
```

Vulnerability Report

Title: [e.g. Reflected XSS in product search]

Severity: [Critical / High / Medium / Low]

Affected asset: [URL / field]

Tested by: [your name]

Summary

[1-2 sentences: what it is and why it matters.]

Steps to Reproduce

1. [step] 2. [step] 3. [step]

Proof of Concept

[the exact payload used]

Impact

[what an attacker can actually do.]

Remediation

[how to fix it.]

Want the full path beyond this week, tools, labs, certs, how to make money? It's all in the roadmap.

See you Thursday. Keep Juice Shop ready.

Workshop 2 practice pack by Mohamed Medjahdi, Cybersecurity Engineer

For educational and cybersecurity awareness purposes only. Only test systems you own or are allowed to.