

LA CTF — web/mutation **mutation** Writeup

Challenge: web/mutation mutation

Event: LA CTF / <https://lac.tf/>

Category: Web

Challenge Author: burturt

Writeup Author: Sunny Nguyen

Tools Used: Chrome DevTools (Mac)

Difficulty: Easy (but tricky if you're new to DevTools)

TL;DR

The page rapidly mutated its displayed content to make the flag hard to read manually. I used Chrome DevTools to **pause execution** from the **Sources** tab, then switched back to **Elements** and copied the flag while the DOM was frozen.

Challenge Description

"It's a free flag! You just gotta inspect the page to get it. Just be quick though... the flag is constantly mutating. Can you catch it before it changes? 🧬"

The site elements constantly changed, making it hard to visually catch the flag before it mutated again.

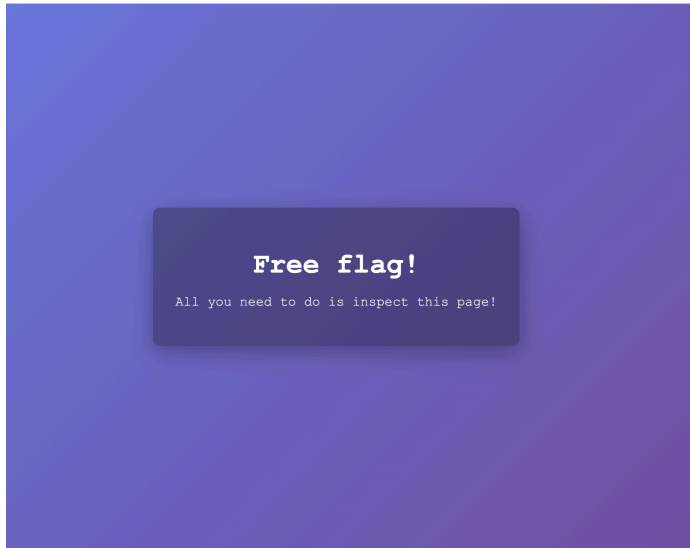
Objective

Capture the flag from a webpage where the DOM content changes rapidly.

Initial Analysis

When I first arrived at the challenge webpage, I noticed two things immediately:

1. **Right-click was disabled**, so I couldn't inspect the page normally.



Since I was on Mac, I opened Chrome DevTools manually using:

- **Cmd + Option + I**

At this point, I could see the page content changing constantly in the Elements panel.

```
Elements  Console  Sources  Network  Performance  Men
<!-- lactf{wrong_flag} 7lwivsaazsbcbrv33dl2inotgd9zsg5 -->
<!-- lactf{fake_flag} rfyf1dag5e4w7rzn8c3oge4hbm237c4 -->
<!-- lactf{not_here} b898cmnyt0rwio2g1zdyjrw93dfc3nfy -->
<!-- lactf{bait} m72sficwb624vl013b3tt57s3wy2j936 -->
<!-- lactf{not_here} 1ju2186drs53whllwpoysty7uv86llpz -->
<!-- lactf{decoy} 214ggwyz90ba50z2oap8285iat0m4bcy -->
<!-- lactf{wrong_one} 1sb7wznnz4m3gd9kndt5ubhjhlnbnttgu -->
<!-- lactf{nice_try} n852lv90cjhj2kux7xucciq2296j7fwr -->
<!-- lactf{nice_try} ykgy6lrqn3cfpte3cba6oluzj6owdvsj -->
<!-- lactf{wrong_flag} j5qfu07zogqf706jzlpfygeo3o5jzd9x -->
<!-- lactf{not_the_flag} 924og07g6b1478udvh3nercrkuaxxju7 -->
<!-- lactf{decoy} rd9e8nc2del0fldy37tlx4hg9zzp1l5 -->
<!-- lactf{bait} zhlcqzvtz0pxilhkf6hj1r0hku505dov -->
<!-- lactf{keep_looking} wpyqw711cxwts10f2ncendkmmzlsge3 -->
<!-- lactf{wrong_flag} vqsc84xdlea1uook3dsfo3592iri9sx7 -->
<html lang="en">
  <head> ... </head>
  <body> ... </body> flex
</html>
```

First idea (not ideal 😊)

My first thought was:

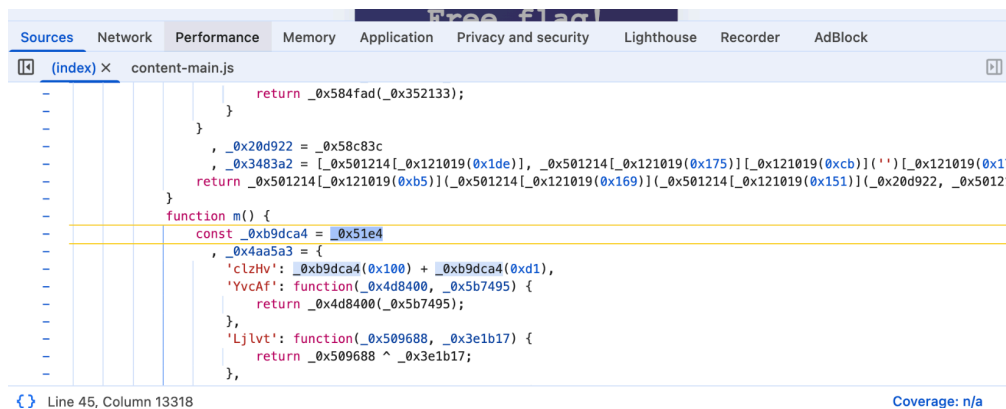
“Maybe I can screenshot the screen and get lucky.”

This would technically work, but it’s unreliable and not really a good technique.

Investigation

I explored Chrome DevTools for a bit (I wasn’t very experienced with it yet), and found the page’s `index` source file.

However, the JavaScript logic was **obfuscated**, so reverse engineering it would’ve taken longer than necessary.

A screenshot of the Chrome DevTools Sources panel. The 'Sources' tab is selected, showing a file named 'content-main.js'. The code is heavily obfuscated, featuring numerous hexadecimal literals and function names. A yellow highlight is placed over a line defining a constant: `const _0xb9dca4 = 0x51e4`. Below this, there are several function definitions and calls, including one for `'YvcAf'` which returns a hexadecimal value. The status bar at the bottom indicates 'Line 45, Column 13318' and 'Coverage: n/a'.

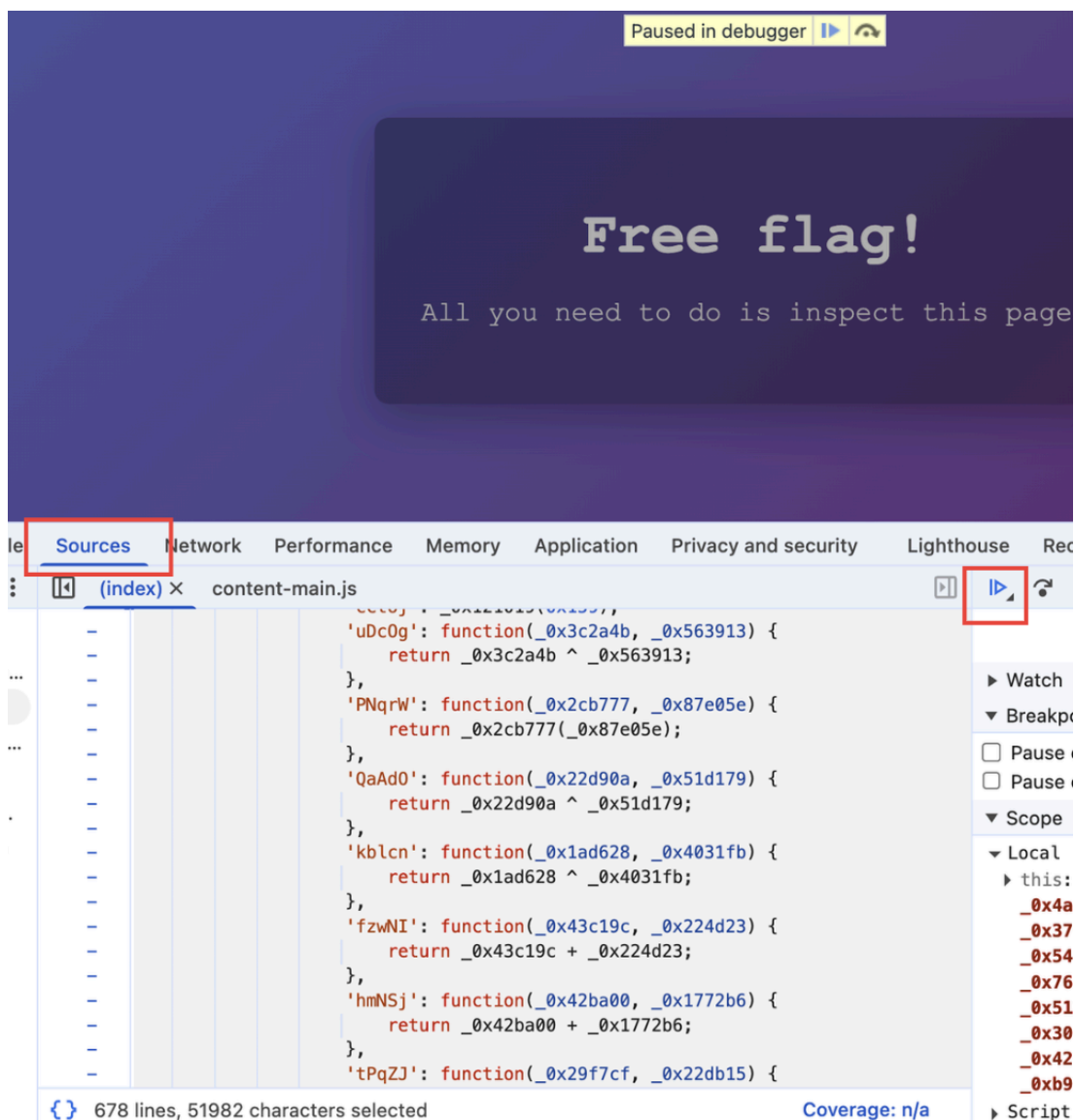
Instead of trying to deobfuscate the script, I looked for a faster approach.

Solution (What Worked)

The simplest solution was to **freeze the page** while the flag was visible.

Steps

1. Open **DevTools**
2. Go to the **Sources** tab
3. Click the **Pause** button (⏏) to pause JavaScript execution



- Because the DOM was frozen while execution was paused, the flag stopped mutating long enough to copy it cleanly.

I was able to successfully copy the flag directly from the frozen DOM.



- Even if right-click is disabled, you can still open DevTools with keyboard shortcuts.
- You don't always need to reverse engineer obfuscated JavaScript — sometimes there's a simpler way.
- **Pausing execution in the Sources tab** is a powerful trick for CTF web challenges where:
 - the DOM is changing rapidly
 - the flag flashes briefly
 - scripts are trying to hide content

Defensive Notes (Real-World Takeaway)

This kind of “security” is not real security — it’s just **client-side obfuscation**.

If a flag (or secret) exists in the client-side DOM at any point, a user can usually:

- freeze the DOM
- intercept responses
- view source
- dump memory/state
- or scrape the page