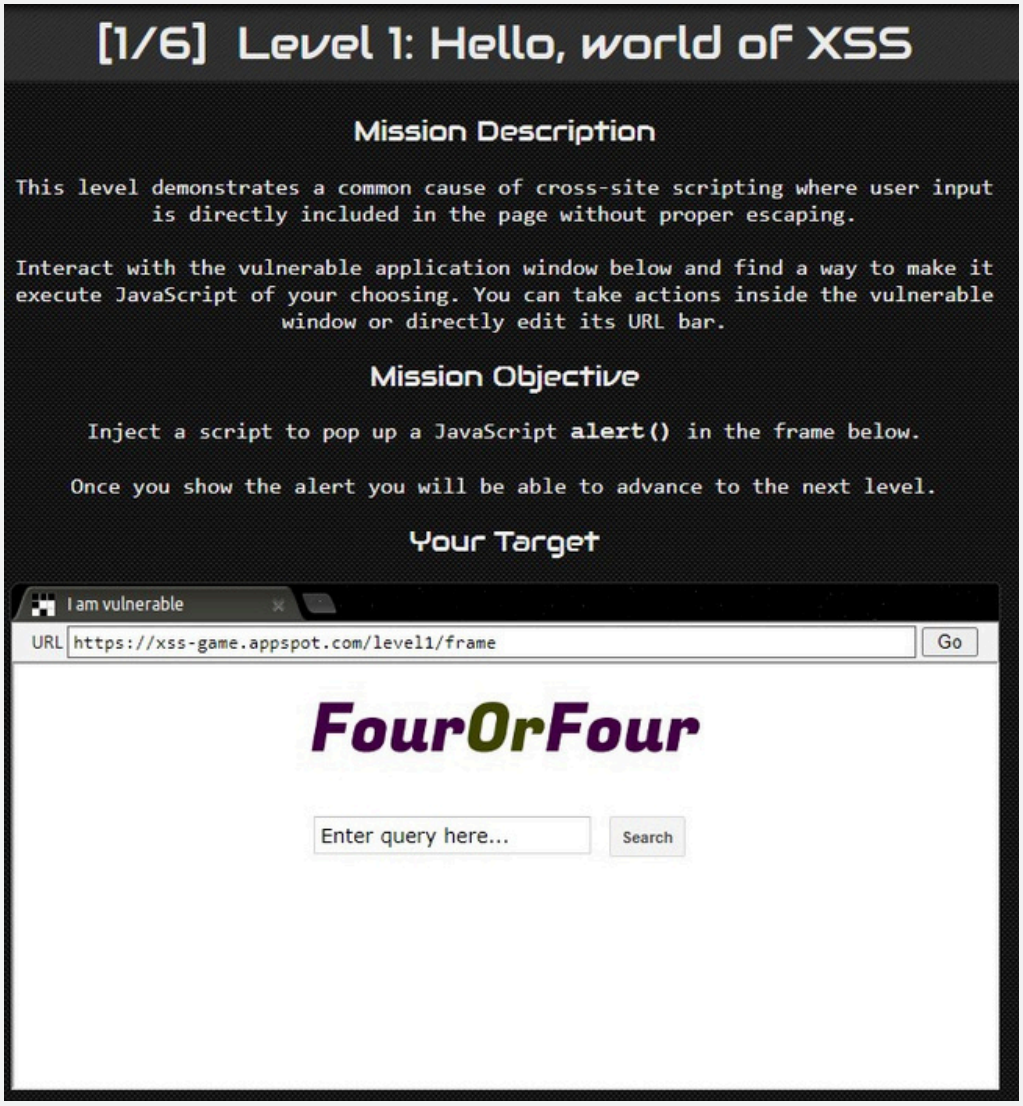
Traccia Bonus
XSS Game

Level 1
XSS Reflective

Procediamo alla visualizzazione del codice sorgente della pagina tramite tasto destro , view page source(oppure consultando semplicemente il source code che ci propone il gioco)



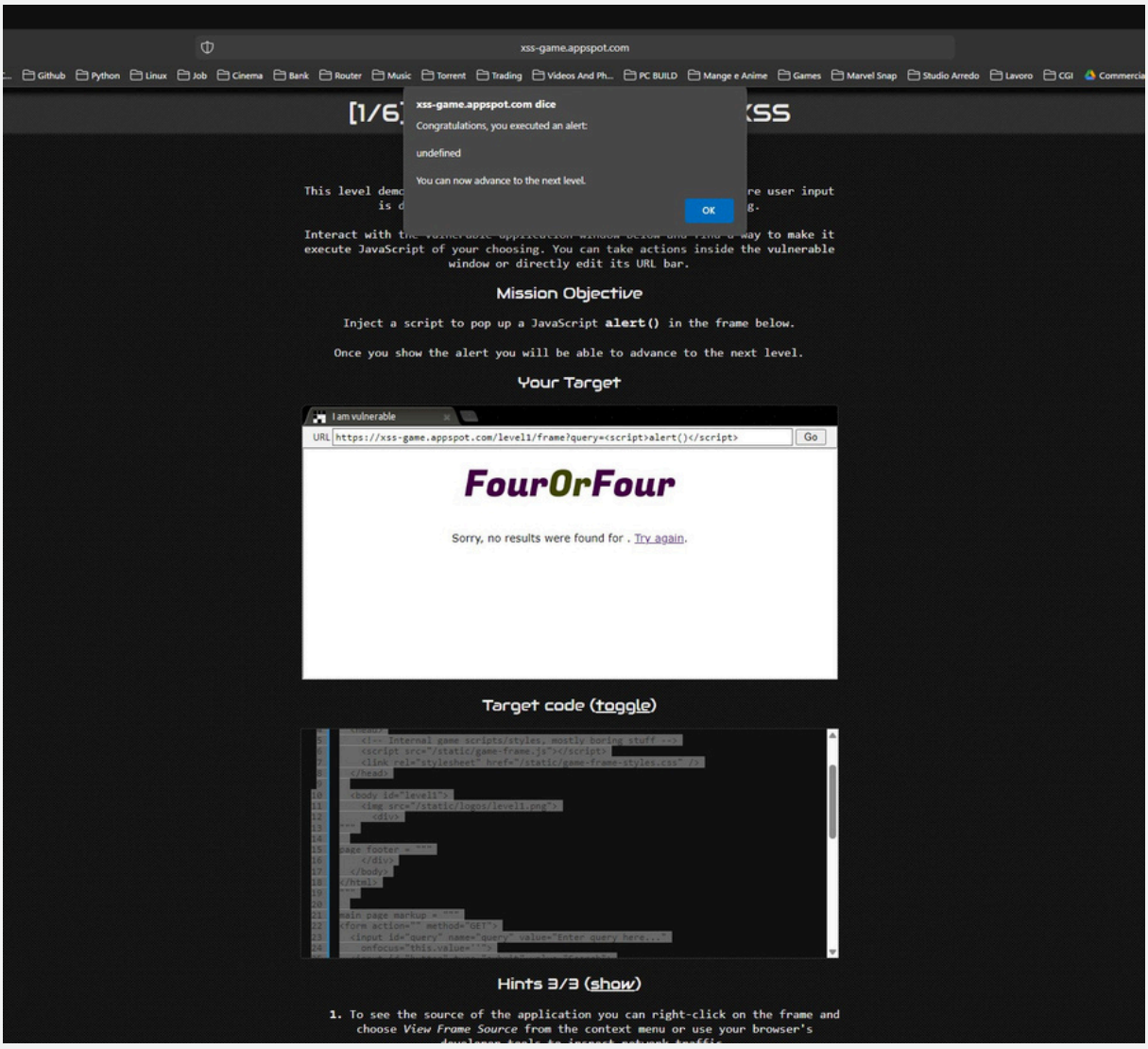
Analizzando il source code possiamo notare la presenza di una vulnerabilità tramite la stringa di codice indicata di seguito:

```
def render_string(self, s):
  self.response.out.write(s)

def get(self):
  # Disable the reflected XSS filter for demonstration purposes
  self.response.headers.add_header("X-XSS-Protection", "0")

  if not self.request.get('query'):
    # Show main search page
    self.render_string(page_header + main_page_markup + page_footer)
  else:
    query = self.request.get('query', '[empty]')
```

La riga evidenziata disabilità il filtro XSS integrato nel browser, che di solito aiuta a prevenire alcuni attacchi XSS riflessi. Disabilitare questo filtro rende l'applicazione più vulnerabile agli attacchi

Proviamo quindi ad inserire all'interno del campo di testo la seguente stringa js:

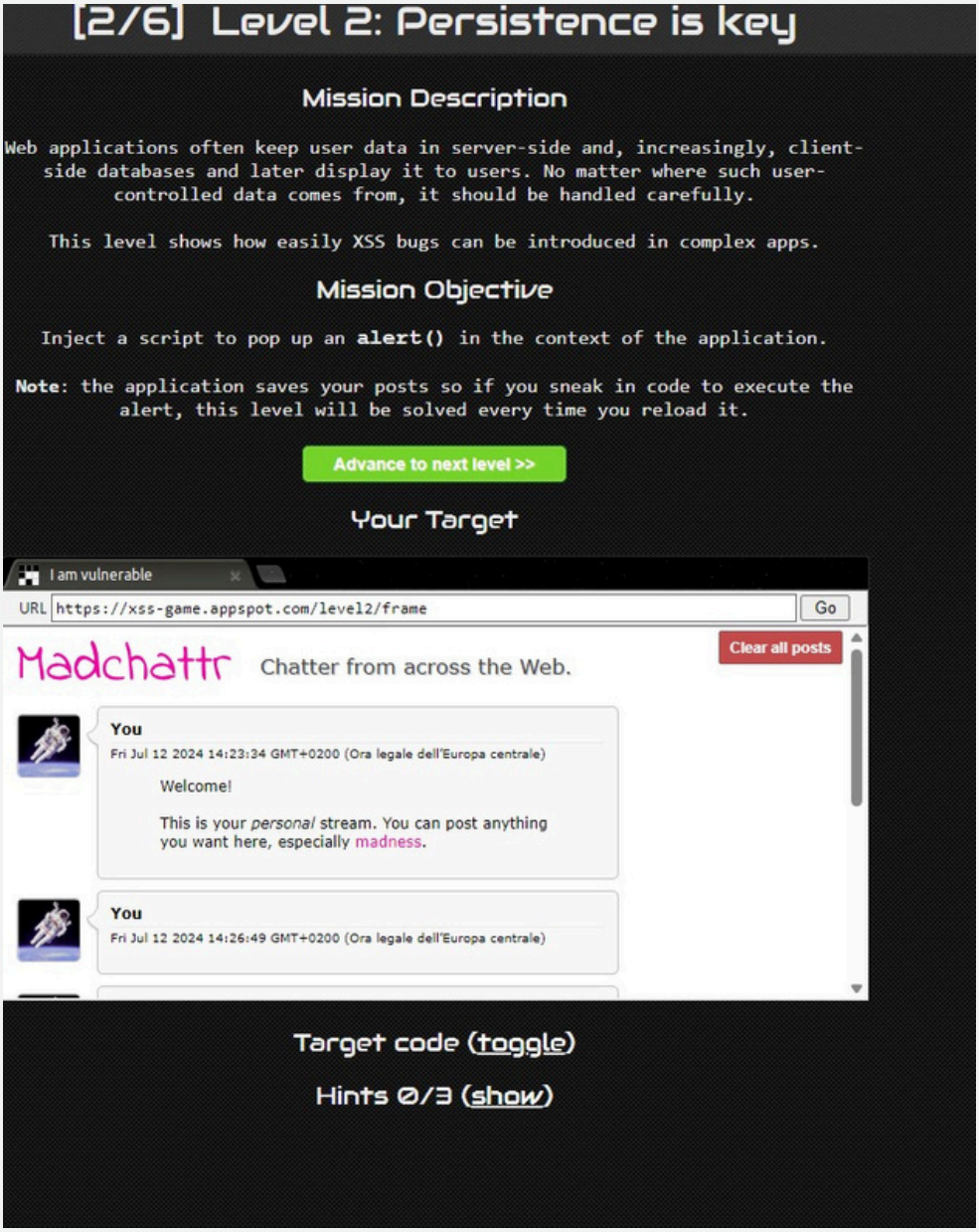
<script>alert()</script>



In questo caso il codice javascript verrà eseguito nel contesto della pagina web , causando l'esecuzione dell'alert.

Level 2
XSS Stored

Nel secondo livello proviamo ad inserire lo stesso codice inserito nel livello 1. Come possiamo vedere non riusciamo a scatenare l'alert con la stessa facilità .



Procediamo quindi ad analizzare il codice sorgente fornito.

Possiamo provare ad iniettare un “onerror” dentro un immagine.
Di seguito il codice inserito:

<img src=”123" onerror=”alert()”>

Creiamo un tag immagine specificando che se l’immagine “123” ci darà un errore(onerror), questo causerà un pop alert. Di seguito il risultato dell’alert: