**JAVASCRIPT**

The attacks in this section are designed to help you learn about how JavaScript is used in the browser and how it can be manipulated. The attacks could be carried out by just analysing network traffic, but that isn't the point and it would also probably be a lot harder.

**Description:**JavaScript is a very famous language used byprogrammers to write scripts for webapplications. The Script can be read and different functions can be called using theconsole to see what the script/function does

**Objective:**

Simply submit the phrase "success" to win the level. Obviously, it isn't quite that easy, each level implements different protection mechanisms, the JavaScript included in the pages has to be analysed and then manipulated to bypass the protections.

**Impact:**If a token is being generated by ascript, we can call that function using theconsole get the token value.

**LOW**

**Steps to reproduce:**

1. Configure your browser and burp suite.

2. Go to the dvwa page and set level of JavaScript to the low level.

3. In source code we can see the token is : md5(rot13(phrase))

4. Write success in the input box and click on the submit and capture the request in the tool like burp suite.

5. Our phrase is success and change it into rot13 and then md5 hash, we get our token value

6. Then change the value of token and forward the request.

**Vulnerable request:**

POST /DVWA/vulnerabilities/javascript/ HTTP/1.1

Host: 127.0.0.1

User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:78.0) Gecko/20100101 Firefox/78.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Content-Type: application/x-www-form-urlencoded

Content-Length: 65

Origin: http://127.0.0.1

Connection: close

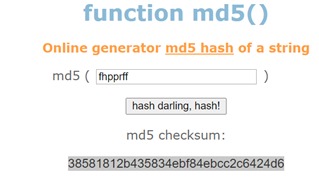
Referer: http://127.0.0.1/DVWA/vulnerabilities/javascript/

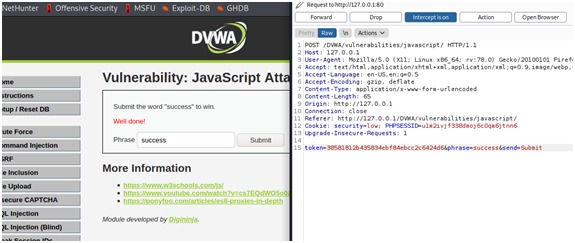
Cookie: security=low; PHPSESSID=u1m2ivjf338dmoj6c0qm6jtnn6

Upgrade-Insecure-Requests: 1

token=8b479aefbd90795395b3e7089ae0dc09&phrase=success&send=Submit



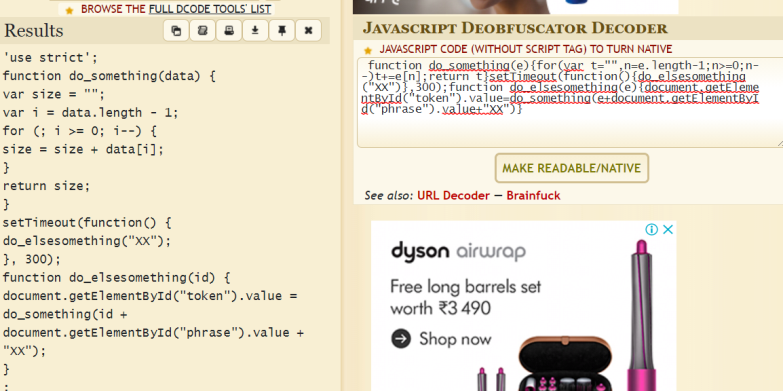




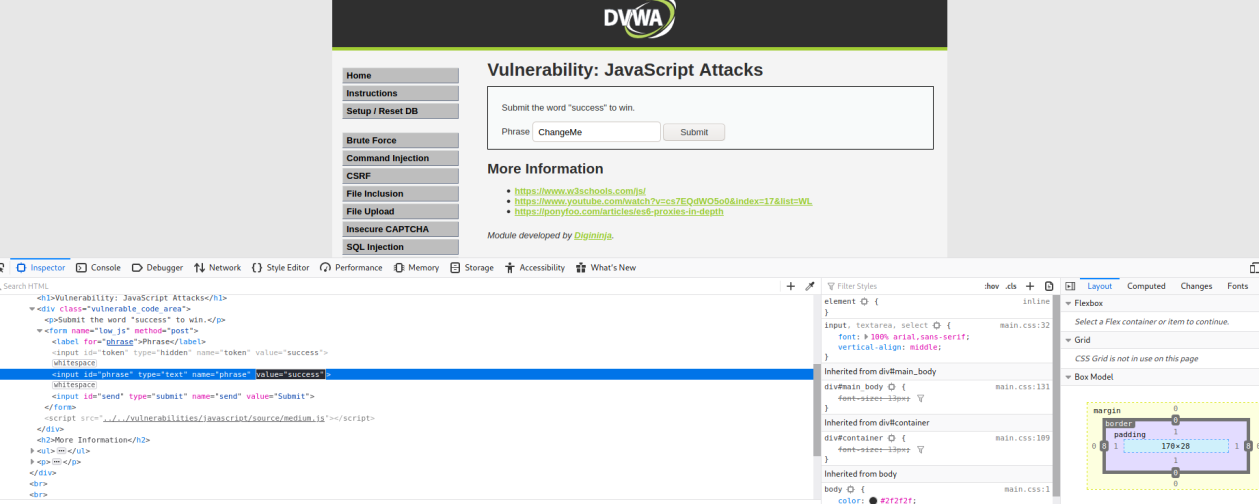
**MEDIUM**

**Steps to reproduce:**

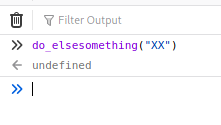
1. Configure your browser.
2. Go to the dvwa page and set level of JavaScript to the medium level.
3. In source code we see the medium.js script , we cannot understand it. So it means that it can be obfuscated. To convert in into readable form.
4. In below screenshot we see that the function that generates the token value runs 300 ms after submitting , So what we have to do is change the value of the token and run the function before submitting .



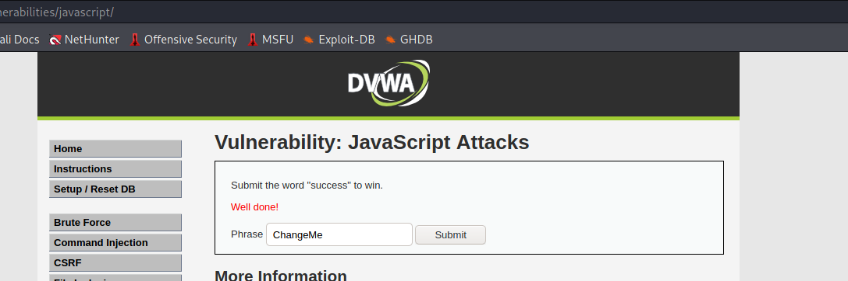
1. Go to the inspect and some value to success screenshot given bellow



1. Now we have to run the function “do\_elsesomething(“XX)” so that the token can be generated. So to that we go to the console tab, type the function and press enter.



1. Now press submit button



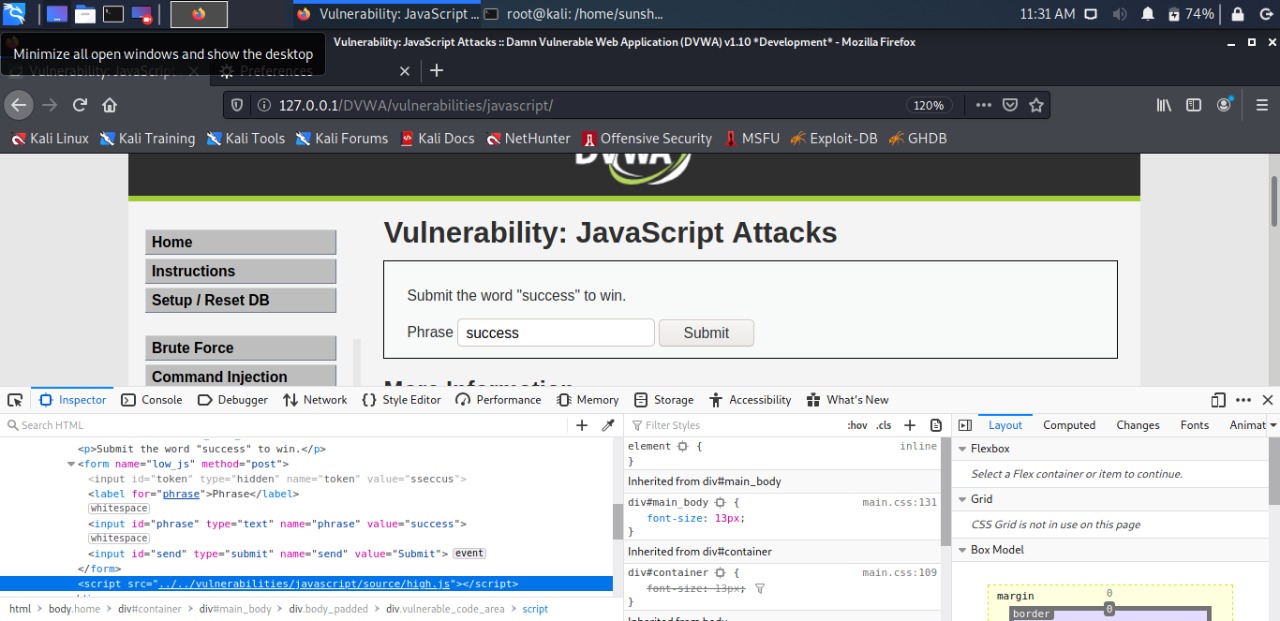
**HIGH**

**Steps to reproduce:**

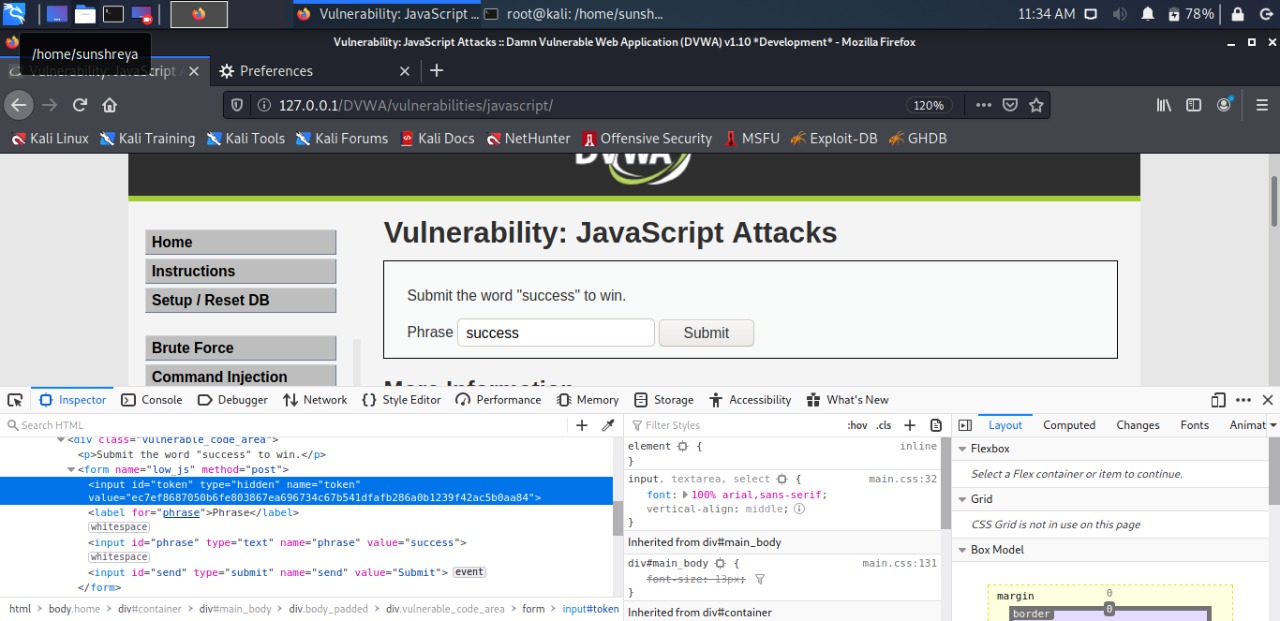
1. Configure your browser.
2. Go to the dvwa page and set level of JavaScript to the medium level.
3. In source code we can see the code is not in readable form so we can so it is obfuscated.



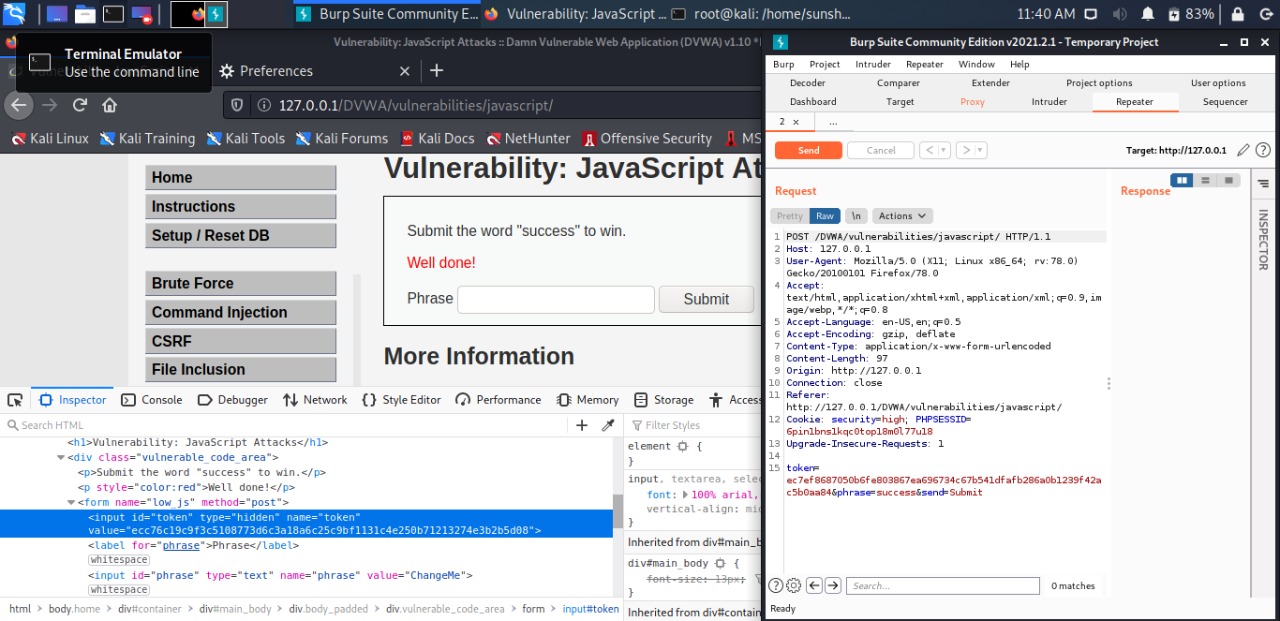
1. Now go to inspect element on the javascript page.
2. Change these values to “success”.



1. Now select the high.js script , right click on it and select “Use in console”.
2. Now we will use all the 3 function to see what changes they make to the phrase. So the first function



1. Copy the value of token.
2. Now write success in input box and capture the request in the tool like burp suite.
3. Change the value token and forward the request.



**Vulnerable request:**

POST /DVWA/vulnerabilities/javascript/ HTTP/1.1

Host: 127.0.0.1

User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:78.0) Gecko/20100101 Firefox/78.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Content-Type: application/x-www-form-urlencoded

Content-Length: 97

Origin: http://127.0.0.1

Connection: close

Referer: http://127.0.0.1/DVWA/vulnerabilities/javascript/

Cookie: security=high; PHPSESSID=6pin1bns1kqc0top18m0l77u18

Upgrade-Insecure-Requests: 1

token=ec7ef8687050b6fe803867ea696734c67b541dfafb286a0b1239f42ac5b0aa84&phrase=success&send=Submit

**Prevention:**

* 1. Source code vulnerability
     1. Source code vulnerabilities may be combined with other—even a number of—JavaScript security holes. In such cases, using a single JavaScript obfuscation cannot prevent or hide these types of vulnerabilities. Because JavaScript is an interpreted, not a compiled, language, it would be virtually impossible to protect application code from being examined by potential hackers with this method. Nonetheless, obfuscation is still a good practice, as it slows down the hackers in their reverse-engineering attempts
  2. Unintended Script execution
     1. The majority of unintended script execution attacks involve [cross-site scripting (XSS)](https://snyk.io/learn/cross-site-scripting/). A particular concern related to JavaScript is the way it interacts with the Document Object Model (DOM) on a web page, allowing scripts to be embedded and executed on client computers across the web. And so, while several different types of XSS attacks exist, what they all have in common is that they cause untrusted script to appear and run in the user’s browser.
     2. For eg: <script>alert(1)</script>
  3. Encoding User Input
     1. XSS attacks rely on supplying data that contains certain special characters that are used in the underlying HTML, JavaScript, or CSS of a web page. When the browser is rendering the web page and encounters these characters, it sees them as part of the code of the web page rather than a value to be displayed. This is what allows the attacker to break out of a text field and supply additional browser-side code that gets executed.
     2. To prevent this, any time browser-supplied data will be returned in a response (whether immediately reflected or retrieved from a database), these special characters should be replaced with escape codes for those characters.