Challenge URL: /dvwa/vulnerabilities/javascript/
Objective: Simply submit the phrase "success" to win the level.
Tools needed: None

Did you remember to read this section's README?

The Guide

Examining the Form

We're presented with a simple form, pre-filled with the string "ChangeMe".

Well, why don't we do the obvious and try submitting "success"?

Hm, no luck. We do get a useful error that states "Invalid token". So that means the form must be defining and setting a token in some manner. Let's examine the source code and see what we can find out. Inspect the form by right-clicking it and selecting "Inspect Element".

Okay, we can notice two things:

1. The form does indeed have a hidden "token" field.
2. Javascript code exists to set "token".

Reverse Engineering the Javascript

If you're anything like me, you saw the Javascript and your eyes started to glaze over. Bit manipulation, obfuscated variable and function names ... it could take hours to reverse engineer.

Hold up though, back the fun bus up a sec. If we examine the code more closely, we see two things that make our job much easier:

1. A commented out line: `MD5 code from here https://github.com/blueimp/JavaScript-MD5`
    - This tells us most of the worst code is simply an implementing of the MD5 hashing algorithm, and can safely be "ignored" for now.
2. Most of the code we care about exists in two comparatively-easy functions, `rot13()` and `generate_token()`.

Let's pull them apart.

`rot13()`

Here's the code, slightly cleaned:

```
1 function rot13(inp) {
2 return inp.replace(/[a-zA-Z]/g,function(c){
3 return String.fromCharCode(
4 (c<="Z"?90:122)>=(c=c.charCodeAt(0)+13)?c:c-26
```

```
5 )
6 }
7 );
8 }
```

Let's discuss the functionality line-by-line.

1. We're defining a function called `rot13()` with one argument, `inp`. We can assume `inp` is our plaintext (ie, what we input in the form), which will be encrypted by the ROT13 cipher.
2. We're replacing any alphabetic character (lowercase or uppercase) in `inp` with whatever `function()` returns.
3. This is where `fromCharCode()` returns, with what it returns being defined in the next line.
4. This is just an implemention of the ROT13 cipher, which "rotates" a single alphabetic character 13 spaces according to the cipher specifications.
5. We're closing out the `fromCharCode()` function definition and returning a single-character output as a string.
6. We're closing out the `function()` function definition, which gives us the replacement single-character string as encrypted by the ROT13 implementation.
7. We're closing out the `replace()` function definition, so we can iterate through all characters in `inp`.
8. We're completing the `rot13()` function definition, so we can return the full ROT13-encrypted result of `inp` to the calling code.

In short, this function takes a string as input and returns the ROT13-encrypted version of it.

`generate_token()`

Here's the code:

```
1 function generate_token() {
2 var phrase = document.getElementById("phrase").value;
3 document.getElementById("token").value = md5(rot13(phrase));
4 }
```

Let's discuss the functionality line-by-line.

1. We're defining a function `generate_token()` that accepts no arguments.
2. We're defining a variable `phrase` and setting it to whatever text we had input in the form.
3. We're setting the value of "token" to our input, once the input has first been encrypted with the previously-defined `rot13()` cipher, then has been hashed with the previously-defined `md5()` hash function.
4. We're completing the `generate_token()` function definition.

In short, this function takes our input, modifies it, and sets "token" to the modified value.

Breaking the Form

So if we've analyzed the code correctly, all we need to do is encrypt the string "success" with the ROT13 cipher. Then we hash the result with the MD5 hashing algorithm. Finally, we set the value of "token" to our hash.

Let's do that step-by-step:

1. rot13("success") = "fhpprff"
2. md5("fhpprff") = "38581812b435834ebf84ebcc2c6424d6"

We can then go back to our form and open up the Firefox developer console by pressing the `Control+Shift+K` keys simultaneously. Input the following line in the console:

```
document.getElementById("token").value = "38581812b435834ebf84ebcc2c6424d6"
```

Then submit the form with the word "success":

We got the success message!

Challenge complete, and DVWA's "low" mode is completely broken!