Challenge URL: /dvwa/vulnerabilities/csrf/
Objective: Make the current user change their own password, without them knowing about their actions, using a CSRF attack.
Tools needed: Burp Suite (optional)

Did you remember to read this section's README?

The Guide

Testing the Form

We'll begin this challenge the standard way, by examining the form and testing its functionality.

Let's try changing our password to "newpw":

Interesting, there's no real verification required. We don't need to know the old password or otherwise confirm that we actually wanted to change it, at least on the form.

Now let's look at the URL, as shown in our browser:

http://dvwa/dvwa/vulnerabilities/csrf/?password_new=newpw&password_conf=newpw&Change=Change#

We don't see encryption or masking of the password in the URL.

If you want to work with Burp, you can also examine the request headers, cookies, or other fun bits. Upon analysis, we don't see much that would hinder a password reset (besides the standard cookie **PHPSESSID**).

I wonder what we can do with this?

The Attack

The attack is actually trivial to implement. All we need to do is craft a new URL with whatever we want the password to be:

http://dvwa/dvwa/vulnerabilities/csrf/?password_new=[your_new_password]**&password_conf=**[your_new_password]**&Change=Change#**

If we can get the "admin" user to visit the malicious URL, the password will be changed to whatever we want. We don't need to actually type in anything to the vulnerable form.

Note the URL in the screenshot, where I changed the password to "hackedpassword" successfully. We can verify by logging out ...

And logging back in with the hacked password ...

It works! Challenge complete.

In the Real World ...

In the real world, when you're conducting a pentest, you'd use social engineering techniques to get a user to click the malicious link for you. Common techniques include spear-phishing emails, IMs, and texts. If the user is authenticated to the vulnerable application, the attack will succeed and the password will be reset without the user knowing. If you want to test this, you'll need to create a new user account that will send the malicious link to the "admin" user via email or a similar method. You'd then want the "admin" user to authenticate to the app and then click the link.

Social engineering attacks are somewhat luck-based, as it relies on the user:

1. Having an active session on the vulnerable app;
2. Getting your message;
3. Reading your message;
4. Clicking the link; and
5. (Ideally) Informing you the link didn't work in the way you pretended it would.

It's never guaranteed to work. That's why I'm not showing you how to do it here. If you want to try it anyway, consider it extra credit :)