

Forum:[Hack Forums](#)

Thread: [\[Tut\] Cross Site Request Forgery \(CSRF\) \[Tut\]](#)

F-Nuke - April 15, 2015, 5:10 am

What is CSRF?

CSRF is Cross Site Request Forgery (sometimes denoted by XSRF, X = "cross"). Don't confuse CSRF with CSS/XSS. The difference between CSRF and XSS is:

In XSS, the trust for a user on the website is exploited whereas in CSRF, the trust that a web server has on a user is exploited.

The whole CSRF exploit revolves around the way the server authenticates a user. It is basically sending out malicious HTTP requests without the user's consent/knowledge.

Lets say this is the HTTP request sent to a web server to transfer money from one account to another account:

PHP Code

```
<?php
POST /csrf/Transfer HTTP/1.1
Referer: http://localhost/csrf
User-Agent: ...
Content-Type: application/x-www-form-urlencoded
Host: 127.0.0.1:54607
Content-Length: xx
Cookie: .PHPSESSID=sjas7dsakjdsa7
Money=10000&toAccount=mike
```

As we can see the POST variables are Money and toAccount. Now lets look at the way, a user is being authenticated by the website:

PHP Code

```
<?php
$username = 'user';
$password = 'pass';
session_start();
$_SESSION['in'] = 1;
$_SESSION['user'] = $username;
```

With that type of authentication, the web server can't be sure if the request was made by the legit user or by a malicious user by the use of CSRF.

Let's take a look at the malicious code the attacker writes to send himself money.

PHP Code

```
<?php
<form name="transfer" method="post" action="http://localhost/csrf/transfer">
  <input type="hidden" name="Money" value="10000">
  <input type="hidden" name="toAccount" vale="john">
```

```
</form>  
<script>document.transfer.submit()</script>
```

The above script submits itself to the transfer page with all the values required. When the legitimate user visits a page with that malicious code in, the form automatically sends a HTTP request to the webapp, basically asking it to send money to the account "john" without the intervention of the user. This happens mostly due to the lack of proper authentication of user submitted requests.

Counter measures:

Create authentication tokens for every request and validate them.

startitit2 - April 15, 2015, 5:20 am

thank you for posting :)
I will see it better

F-Nuke - April 15, 2015, 5:21 am

(Apr 15, 2015 05:20 AM)startitit2 Wrote: [\[CITE\]](#) thank you for posting :)
I will see it better

Sure! Let me know if you have any doubts or if you want me to add something to it.

acidgod21 - April 16, 2015, 9:17 pm

omg lookin 4 dis everywhere, ur a lifesavor bro! #thx

HelpingOthers - April 16, 2015, 10:50 pm

Thanks I have bookmarked this tutorial for reference

F-Nuke - April 21, 2015, 4:40 am

(Apr 16, 2015 10:50 PM)HelpingOthers Wrote: [\[CITE\]](#) Thanks I have bookmarked this tutorial for reference
Thanks for the feedback! Glad you like it.

Vector.A+ - May 1, 2015, 10:10 am

thank you a very nice tutorial

F-Nuke - May 1, 2015, 2:05 pm

(May 1, 2015 10:10 AM)max_steel Wrote: [\[CITE\]](#) thank you a very nice tutorial
Thanks for the feedback! Glad you like it.

Water Aura - June 4, 2015, 12:07 pm

Good tutorial, but slightly useless in most cases.

Many people use anti-csrf tokens (verification of cookie and input value) to prevent such a thing from EVER occurring.

F-Nuke - June 4, 2015, 12:43 pm

(Jun 4, 2015 12:07 PM)Water Aura Wrote: [\[CITE\]](#) Good tutorial, but slightly useless in most cases.

Many people use anti-csrf tokens (verification of cookie and input value) to prevent such a thing from EVER occurring.

I agree that it is not an effective vulnerability. But not many people use anti csrf tokens. Only big sites do.

TheRoger - *June 4, 2015, 6:38 pm*

bookmarked, thank you for this guide :)