Cross Site Request Forgery

Posted At : June 27, 2008 5:41 PM | Posted By : Nathan Mische
Related Categories: ColdFusion

Last week I attended a web application security workshop presented by the SANS Institute. While a majority of the content was a review for me, I did learn about one type of attack I was not familiar with, the Cross Site Request Forgery (CSRF) attack. This type of attack, also known as session riding, has been around for awhile and is really pretty simple.

The Attack

In order for this attack to work the attacker knows the URL or form submission needed to perform some action on the target site. For example, let's say they know http://www.myapp.com/index.cfm?action=delete&id=123 deletes an item from your application. In order to perform this action the user needs to be logged in to your application, and this is where the attack comes into play. Assume the victim is logged into your application and the attacker sends the victim an e-mail or otherwise gets the user to browse to a site controlled by the attacker. That email or site contains content, say an image tag or 1 pixel iframe, that makes a request to the above URL. This will cause the above action to be taken, likely without the victims knowledge, because request for the image or iframe content is made from the victims browser and thus includes their session cookies.

The Defense

There are several things you can do to mitigate the risk of a CSRF attack including allowing only the POST method to trigger actions in your application, making sure your your application has a session timeout, and checking the HTTP Referer. These defenses only make the attack slightly more challenging. The strongest defenses are CAPTCHA or the use of an anti-CSRF token. I think most people are familiar with CAPTCHA, but what is an anti-CSRF token? Basically it is a random token, dynamically generated for each form, that is then attached to the form and verified on submittal. It works because in a CSRF attack the attacker doesn't see the content being sent from the server to client, and thus would have to guess at the token. There are several approaches you could use in generating the token, but the thing to keep in mind is that it has to be verifiable after form submission. Given that, one approach which I'll demonstrate below is to generate a token based on the hash of the form name, the user's session ID and a secret key. Here is an example:

```
<cfapplication name= "CSRF"  sessionmanagement= "true">

<cfsilent>

    <cfset formName =  "exampleForm" />

    <cfset secretKey =  "you will never guess" />

</cfsilent>


<!DOCTYPE html PUBLIC  "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html xmlns= "http://www.w3.org/1999/xhtml">

    <head>

        <title>CSRF Defense </title>

    </head>

    <body>


    <cfif IsDefined('form.submit')>


        <!--- here we check the token to make sure the form submittal is valid --->

        <cfif IsDefined('form.token') and form.token eq Hash(formName & session.sessionID & secretKey)>

            <cfoutput> <p>Welcome #form.fName# #form.lName#.  </p></cfoutput>

        <cfelse>

            <p>Invalid form submission. </p>

        </cfif>


    <cfelse>


        <cfoutput>

        <form action= "CSRF.cfm"  method= "POST">
```

```
            <fieldset>
                <label for="fName">First Name:</label><input type="text" name="fName" id="fName" /><br>
                <label for="lName">Last Name:</label><input type="text" name="lName" id="lName" /><br>
                <input type="submit" name="submit" />
                <input type="hidden" name="token" value="#Hash(formName & session.sessionID & secretKey)#"
            </fieldset>


        </form>
        </cfoutput>


    </cfif>


    </body>
</html>
```

Obviously the need to mitigate this type of risk depends on the application, but it is certainly something to be aware of. To be honest I was a little surprised that I wasn't familiar with this attack given how simple it is.