veracode.com

Cross-Site Request Forgery Guide: Learn All About CSRF Attacks and CSRF Protection

5-6 minutes

Cross-Site Request Forgery (CSRF) is an attack outlined in the OWASP Top 10 whereby a malicious website will send a request to a web application that a user is already authenticated against from a different website. This way an attacker can access functionality in a target web application via the victim's already authenticated browser. Targets include web applications like social media, in-browser email clients, online banking and web interfaces for network devices.

Key Concepts of Cross-Site Request Forgery

- Malicious requests are sent from a site that a user visits to another site that the attacker believes the victim is validated against.
- The malicious requests are routed to the target site via the victim’s browser, which is authenticated against the target site.
- The vulnerability lies in the affected web application, not the victim’s browser or the site hosting the CSRF.

Executing a CSRF Attack

In a Cross-Site Request Forgery attack, the attacker is exploiting how the target web application manages authentication. For CSRF to be exploited, the victim must be authenticated against (logged into) the target site. For instance, let's say *examplebank.com* has online banking that is vulnerable to CSRF. If I visit a page containing a CSRF attack on *examplebank.com* but am not currently logged in, nothing happens. If I am logged in, however, the requests in the attack will be executed as if they were actions that I had intended to take.

Let's look at how the attack described above would work in a bit more detail. First, let's assume that I'm logged into my account on *examplebank.com,* which allows for standard online banking features, including transferring funds to another account.

Now let's say I happen to visit *somemaliciousssite.com*. It just so happens that this site is trying to attack people who bank with *examplebank.com* and has set up a CSRF attack on its site. The attack will transfer $1,500.00 to account number 123456789. Somewhere on *somemaliciousssite.com,* attackers have added this line of code:

<iframe src="//www.veracode.com/%3Ca%20href%3D"http://examplebank.com/app/transferFunds?amount=1500&destinationAccount=123456789">http://examplebank.com/app/transferFunds?amount=1500&destinationAccount=..." >

Upon loading that iframe, my browser will send that request to *examplebank.com,* which my browser has already logged in as

me. The request will be processed and send $1,500.00 to account 123456789.

Another Example of Cross-Site Request Forgery

I just bought a new home wireless router. Like most wifi routers, it's configured through a web interface. The router was shipped to me with an internal IP address of 192.168.1.1. I'm having trouble configuring the router though, and fortunately the folks over at *somemaliciousssite.com* have published a guide that shows me exactly what buttons to click in the router interface to get everything set up securely. The attackers have also set up a proxy server at 123.45.67.89 that will log all traffic that goes through it and look for things like passwords and session tokens.

As I clicked through the configuration guide, I missed the 1x1 pixel image that failed to load:

**

The attackers knew that when I was reading their tutorial, I would be logged into the router interface. So they had the CSRF attack set up in the tutorial. With that request, my router would be reconfigured so that my traffic will be routed to their proxy server where they can do all manner of bad things with it.

Preventing Cross-Site Request Forgery (CSRF) Vulnerabilities

The most common method to prevent Cross-Site Request

Forgery (CSRF) attacks is to append CSRF tokens to each request and associate them with the user's session. Such tokens should at a minimum be unique per user session, but can also be unique per request. By including a challenge token with each request, the developer can ensure that the request is valid and not coming from a source other than the user.

Finding and Remediating Cross-Site Request Forgery (CSRF) Vulnerabilities

The easiest way to check whether an application is vulnerable is to see if each link and form contains an unpredictable token for each user. Without such an unpredictable token, attackers can forge malicious requests. Focus on the links and forms that invoke state-changing functions, since those are the most important CSRF targets.

More Security Threat Guides from Veracode

- LDAP Injection
- Mobile Security
- Cross-Site Scripting Vulnerabilities
- Prevention of SQL Injection