Bright

Resource Center > Blog

CSRF tokens: What is a CSRF token and how does it work?

CSRF (Cross Site Request Forgery) tokens can be a great mechanism in preventing CSRF attacks, but what are they? How do they protect against CSRF attacks? How should they be generated? We are going to cover the answers to these and more questions in this blog post.

In this article you are going to learn:

What is a CSRF token?

A CSRF Token is a secret, unique and unpredictable value a server-side application generates in order to protect CSRF vulnerable resources.

The tokens are generated and submitted by the server-side application in a subsequent HTTP request made by the client.

After the request is made, the server side application compares the two tokens found in the user session and in the request. If the token is missing or does not match the value within the user session, the request is rejected, the user session terminated and the event logged as a potential CSRF attack.

How should CSRF tokens be generated?

Just like session tokens in general, CSRF tokens should contain significant entropy and be strongly unpredictable.

You can achieve this by using a cryptographic strength pseudo-random number generator (PRNG), seeded with the timestamp when it was created and a static secret.

For further security, you can generate individual tokens by chaining their outputs with user-specific entropy and take a strong hash of the whole structure.

This presents an additional obstacle to a malicious user who attempts to analyze the tokens based on a sample that is issued to him.

In short, here are the principles you should follow when generating and verifying your token:

- Use a well-established random number generator with enough entropy
- Make sure tokens can't be reused. Expire them after a short amount of time
- Verify the received token is the same as the set token in a safe way, for example, compare hashes
- Do not send CSRF tokens in HTTP GET requests. This will make sure they are not directly available in the URL and they don't leak in the Referer header with other referrer information

For example, a CSRF token in PHP can be generated as follows:

```
$_SESSION['token'] = bin2hex(random_bytes(24));
```

And verify the token as follows:

```
if (hash_equals($_SESSION['token'], $_POST['token'])) {
// Action if the token is valid
} else {
// Action if the token is invalid
}
```

If you prefer a more secure approach, generate separate tokens for each form. Make sure you don't expose the token directly to the user's browser. Hash the token using the filename of the form Hash the token with the filename of the form. Here is one example how to do it in PHP:

```
hash_hmac('sha256', 'post.php', $_SESSION['internal_token'])
```

When you verify, compare the hashes. If both the token and the form are valid, the hashes will match.

How should CSRF tokens be transmitted?

CSRF tokens are secrets and should be handled as such in a secure manner throughout their lifecycle.

Try transmitting the token to the client within a hidden HTML form field, using the POST method. This way the token will be included as a request parameter when the form is submitted:

[example]

Place the field containing the CSRF token as early as possible within the HTML file. Place the field that contains the token before any non-hidden fields and before any places where user-controllable data is embedded. This way you will mitigate the risk against various techniquest where an attacker can use crafted data to manipulate the HTML document and capture part of its content.

For example:

```
<form action="/transfer.do" method="post">
<input type="hidden" name="CSRFToken" value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEw
YzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZMGYwMGEwOA==">
[...]
</form>
```

You can place the CSRF token into the URL query string, but this approach is less safe, as the query string:

- Is logged in various locations (client and server-side)
- Can be transmitted to third parties within the HTTP Referer header
- Can be displayed on-screen within the user's browser

To further defend against an attacker who manages to predict or capture another user's token, insert the CSRF token in the custom HTTP request header via JavaScript. This approach is particularly well suited for AJAX or API endpoints. Browsers usually don't allow custom headers to be sent cross-domain. The downside of this approach is the limitation for the application to make CSRF-protected requests using XHR, and might be considered over-complicated for many situations.

Never transmit CSRF tokens within cookies.

How to fix the CSRF vulnerability in popular web frameworks?

CSRF protection in Angular

Angular is a popular frontend framework developed by Google. It's an open-source project and offers its own set of user interface components that work across devices and platforms.

Angular packs the common security measure of reading the CSRF token called “CSRF-TOKEN”, and sets a custom header named “X-XSRF-TOKEN”. However, Angular is just a client-side framework, so to protect against CSRF, your web server must support this protective method as well.

CSRF protection in Django

Django is a free backend framework based on Python. Django focuses on reusability of code and pluggability of modules along with low coupling and rapid development principles.

Django offers middleware for protecting a web server against CSRF attacks. To protect your apps, the middleware must be activated in your project. Also, you have to include the csrf_token tag inside the form elements which point to any in-project URLs.

CSRF protection in Express

Express is a backend web framework for Node.js. It is fast, flexible and minimalistic. It’s free and open source.

Since Express is a minimalistic web framework, it doesn’t support any anti-CSRF measures by default. But it provides a pluggable middleware that helps your web server to protect itself against CSRF attacks.

The middleware is known as “csurf”, and it’s super easy to set up in your project. It offers some bootstrap options as well to configure its functionality.

CSRF protection in Laravel

Laravel is a free, open source web framework for PHP. Laravel supports a modular packaging system and offers numerous utilities to ease the development and maintenance of web applications.

Part of Laravel’s middleware group is middleware named VerifyCsrfToken. VerifyCsrfToken auto-verifies the token in incoming web requests and disregards CSRF-based requests. To use it, just include @csrf in your forms to include the token field.

CSRF protection in React

React is a front-end framework developed by Facebook. It’s free and open source and is mostly used for building mobile or single-page applications.

Unlike Angular, React doesn’t come with a default security measure against CSRF attacks.

In order to protect a React application against CSRF, you have to introduce a security solution in your app, and have the web server support it.

Luckily, it’s easy to implement CSRF protection in React. You only have to store the CSRF token in your React app and generate relevant headers to send along with the request to the server. The server will quarantine all CSRF requests.

Bright's technology for CSRF Token security

You can easily stop CSRF attacks by just adding code that requires a CSRF token. To do so, you need to know which applications are vulnerable and where, and Bright can help!

Bright automatically scans every aspect of your apps, providing actionable reports. It seamlessly integrates with the tools and workflows developers already use. Scans are fast as our AI-powered engine can understand application architecture and generate sophisticated and targeted attacks. The results are completely false-positive free, so you can focus on releasing code.

Try Bright today, and start testing for CSRF, XSS, and hundreds of other vulnerabilities in minutes – https://app.brightsec.com/signup

Want to learn more about CSRF?

Have a look at these articles:

CSRF Attacks: Real Life Attacks and Code Walkthrough
What is a Cross-site Request Forgery (CSRF) attack?
CSRF vs XSS: What do they have in common and what is the difference

Publication:

June 11, 2021

Writer:

Admir Dizdar

Related Articles:

Four Ways AI Poses a Threat to Cybersecurity and How to Protect Yourself

What is SASE, where is it going, and why does it matter?

Security Breaches: What We Learned in 2022

Web Application Testing: Tips & Best Practices

Deserialization Vulnerability: Everything You Need to Know

7 SSRF Mitigation Techniques You Must Know

   

EXPLORE OUR DAST

BOOK A DEMO

   

Resources

Blog
Docs
Upcoming Events
Videos
Success Stories
News

Product

Product

Company

About Us
We Are Hiring!
Bug Bounty Program
Security

Legal

Terms of Use
Privacy Policy
Cookies Policy