RequestVerificationToken does not match

Asked 13 years, 2 months ago Modified 3 years, 3 months ago Viewed 49k times

35

I have a problem with the anti CRSF MVC mechanism. The cookie and the form input returned does not match. I'm getting an error every single time, only in one specific page. In the rest of the application it works well.

The server is returning `HTTP 500 Internal Server Error` and I can see on the log this exception:

> [System.Web.Mvc.HttpAntiForgeryException]: {"A required anti-forgery token was not supplied or was invalid."}

This is the hidden input that the server is generating is:

```
<input name="__RequestVerificationToken" type="hidden"
value="QK8P7rjyZE6Vm5seY7Fr704YCOoFGdTIMzl1W7R0ZFpXSMjGKLG2T05DfFSYTxvtQCEx7DDT69D
```

And this is the Cookie returned:

```
Set-
Cookie:__RequestVerificationToken_L2VGbG93=skmTAVI8HCbfxDS+xhioIMIISL3UOBI7qJM1JbH
path=/; HttpOnly
```

When I examine what the server is sending, the cookie is exactly the same, but the payload has different encoding I think:

```
__RequestVerificationToken:QK8P7rjyZE6Vm5seY7Fr704YCOoFGdTIMzl1W7R0ZFpXSMjGKLG2T05
```

The differences are in two characters that appear encoded:

```
    /    ->   %2F
    +    ->   %2B
```

Those are the only differences I can find between the hidden input field, and the post payload.

What could be the problem that is causing that `ValidateAntiForgeryToken` fails in verify the token?

Regards.

asp.net-mvc asp.net-mvc-3 security csrf antiforgerytoken

Share
Improve this question
Follow

edited Oct 14, 2011 at 18:48
Charles
51.4k ● 13 ● 106 ● 143

asked Oct 14, 2011 at 11:31
vtortola
35.8k ● 31 ● 167 ● 268

The encoding will not affect the processing of the token and cookie. Please provide details about the page with the issue, and the action being used to transmit the request to your action. In addition (first really), inspect the failing request with Fiddler or a similar tool, and confirm that the token and cookie are both being transmitted. – counsellorben Oct 14, 2011 at 11:57

I can confirm that both the cookie and the input are being sent to the server. – vtortola Oct 14, 2011 at 13:33

Can you please provide details about the view and the action where you are having the problem. The most likely issue would be if you are using a salt, and the salts don't match, but there is no way to diagnose this if you won't provide details. – counsellorben Oct 14, 2011 at 13:36

How are you calling the action? Are you using AJAX? – Darin Dimitrov Oct 14, 2011 at 13:42

yes, it is a jQuery post call. I can see in the Chrome inspector that the call has the cookie and is sending the post payload correctly. – vtortola Oct 14, 2011 at 14:13

3 Answers

Sorted by: Highest score (default)

80

+50

I've had and resolved several issues with `ValidateAntiForgeryToken` lately, so I'll share my findings with you.

Salt: Since you mention this only happens on a single page, my best guess is that you are using different `salt` values in your calls to `Html.AntiForgeryToken(salt)` and `ValidateAntiForgeryToken(salt)` calls.

AJAX: as another answer has said, using AJAX may require extra work to ensure the token is included in the POST. Here is my favorite simple, automatic solution to add the token to all AJAX POST requests.
In your question though, you state that you have verified that the token is sending. Have you verified that you're only sending the token once? I found out that an AJAX call of mine was sending the token twice, which combined the values, and caused it to fail.

Machine Key and Cookies: this issue is ugly, easy to spot (causes exceptions), but not very intuitive. The validation cookies and tokens are encoded and decoded using

a unique "machine key". This means that if you have a server farm, or change your server, your cookie will no longer be valid. Closing your browser fixes the issue (because the cookie is a session cookie). However, some people leave their browser windows open in the background for a long time!
The solution is to set a "machine key" in your config file. This will tell MVC to use the same key on all servers, ensuring that the cookie will be decryptable everywhere.

Encoding Bugs: using a testing utility called jMeter, we attempted to load-test our pages, only to find out that it had a bug that caused our token to have 2 extra `"` around the value.
The solution is to lower your trust in your tools! Test in a browser, and if that works, create a test that extracts the token and cookie values, and set a breakpoint to verify the results.

If none of these things work for you, then I'd recommend taking a look at the MVC source code for `ValidateAntiForgeryTokenAttribute`, specifically the `OnAuthorization` method. It will help you see the different steps where validation could fail. You might even inspect your error's `Exception.StackTrace` to determine which part is failing.

As a side note, I really dislike the implementation of `ValidateAntiForgeryToken` in MVC, because:

- There are about 5 verification steps that can fail, but there is only one generic error message.
- The class is sealed, so it cannot be extended with additional functionality.
- The encryption method is weird - it initializes a `Page` and creates an artificial `ViewState` to encrypt the tokens and cookies. Seems overkill.

So, I grabbed the source code, and created my own specialized subclass, which also turned out to be very helpful in debugging its issues, because I could set breakpoints on the validation methods, and it was really easy to determine which validation step was failing.

Share
Improve this answer
Follow

edited Nov 7, 2014 at 18:02

community wiki
7 revs
Scott Rippey

1 I'm not using salt, and I'm only using 1 server. As per OP, I noticed the Encoding, but figured it was a non issue. Which leaves the AJAX implementation that I've used. I'm using jquery post, and form.serialize. Fiddler confirms that the request is sent as a standard form post and has a seperate parameter for _RequestVerificationToken – stevenrcfox Oct 24, 2011 at 10:04

1 Well then, more information is definitely needed to determine the issue. Source code would be helpful, especially the jQuery part. I also edited my answer to include a link to the

`ValidateAntiForgeryToken` source code, which might give you a better understanding of what's going on. – Scott Rippey Oct 25, 2011 at 5:22

1 Ok, I still dont believe its the validation thats the issue. I think its the creation. The Token in the cookie and the form field are different, not just encoding different. They are also both set by the same response as seen via fiddler,so its not an issue with a partial view. – stevenrcfox Oct 25, 2011 at 10:28

9 Actually, the cookie and token are always different. The cookie remains the same between requests, and the token changes every time it is rendered. However, they always decrypt to the same value. – Scott Rippey Oct 26, 2011 at 3:37

Bounty awarded on the basis of best answer thus far, and definate educational value. Still can't figure out the issue, will post a link to a demo illustrating the problem – stevenrcfox Oct 27, 2011 at 14:09

4

If this is being sent as an Ajax request, then the current setup of the framework isn't build to do this naturally.

Luckly Phil Haak wrote a nice blog post on dealing with CSRF and Ajax -> Preventing CSRF With Ajax which goes into some good detail about how to use the existing framework and modify it to work for Ajax/Json.

Share Improve this answer Follow

answered Oct 21, 2011 at 9:18

Stephen
1,737 ● 2 ● 26 ● 38

1 Just to clarify, typical AJAX requests (posting a form) should work fine with the `AntiForgeryToken` ... it's mostly JSON calls or custom `data` attributes that need extra consideration. See this GitHub gist for a better solution than Phil's. – Scott Rippey Oct 24, 2011 at 6:18

1 Sent as a form post, looks good with Fiddler. The response that actually *sets* the cookie and form value have different tokens though. – stevenrcfox Oct 25, 2011 at 10:31

2

From my recent findings ...

If you set content type as "application/x-www-form-urlencoded" in the ajax request then you must put the AFRT in the data

If you set the content type to "application/json" then the token goes in the ajax "headers" property as described by haack.

On the server if you are checking for the form type token then using the vanilla AntiForgeryRequestTokenAttribute is ok but if you want to validate tokens sent in the header then you need to call the AntiForgeryToken.OnAuthorize ... or whatever, passing the token from the cookie (http context).

It aint easy but if it was everybody would be doing it :)

Share Improve this answer Follow

answered Aug 1, 2013 at 10:06

ComeIn
1,609 ● 19 ● 13

there is no need of token when using application/json. You cannot perform XSRF with JSON, the SOP won't allow the browser to post JSON to another domain. – vtortola Aug 1, 2013 at 15:13