A Frank Discussion About Protection

Posted At : June 19, 2013 1:52 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Security, ColdFusion

I know it's an uncomfortable topic. I understand that you would like to keep your validation private. You would probably rather learn about this from your friends at the coffee shop, Jeremy who is two cubes down from you, or some guy on a forum (shudder). Still, the Muse has an assignment in life to point these things out and make sure you are well informed and prepared when temptation strikes. Oh I know what you say now. I know what I'm doing. The risk factor is slight. I'm too small... I mean... my application is too small to need it. But take it from me - you will need to understand how to use protection or bad things will happen. So let's talk about it.

The Risk Factor

Who's at risk? Simple, if you use form, cookie or URL variables you are at risk. And honestly who doesn't use form, cookie and URL variables? To paraphrase Freud, "The only abnormal user input is *no* user input." I know you think your users are safe, but it only takes one successful injection attempt and you are infected. So it is important to recognize that protection is always necessary in every case and should be a normal part of your development life. The world is full of users who are less than competent - not to mention bots and agents who will show up for one or two HTTP requests and then leave without saying a word in the middle of the night (don't get me started!). So no matter how innocent your application is and no matter how gently used - you still need protection.

What Is "Validation"

I could spend a lot of time talking about technique and how to communicate with your users and if you should use this brand of protection or that - but enough with the uncomfortable metaphors. The important thing to remember is that *anything that comes from the user should be validated on the server*. Did I mention "on the server". I would like to start a campaign to eradicate the use of "client side" and "security" in the same sentence. You see it's not enough to add fancy jQuery validation libraries or any other client side technology. Let me put it to you this way - *there is no client side technology that cannot be circumvented* - usually by the youngest member of the high school computer science club (probably even Jr. High). And there is nothing you can do to prevent them from trying. Why?

Why Client Side Validation is Useless for Security

Let's say you have a form on your web site. The form submits a shopping cart for final processing. You have worked hard to obscure the inner workings of your form. You use hidden form fields. You generally make your internal code look like a JavaScript playground. Your amounts, taxes, quantities, CC information, contact information etc. - all come with nice checks that tell the user useful things like "That is not a valid email" or "The letter 'A' is not a quantity" or "MC Cheese Whiz is not an appropriate name for a beneficiary." You have made a tremendous effort to insure that it works in all browsers. By the time the form is actually posted to the server you are satisfied that it contains what you want it to contain - right? Not exactly...

You see the problem is that nothing prevents a user - a malicious user or even one who is as dumb as a coal bucket - from tampering with your precious client side code. Using

a proxy like **Charles** a novice user could trap all the form field names and values posted to the server, dummy up a new form with those field names, and start posting all sorts of bad data. Of course that's a lot of work. Some tools and plugins allow you to simply manipulate form data directly prior to posting it - rewriting your JavaScript in the process. There is *no* technology that exists on the client side that cannot be tampered with. That's the essence of the client side. It is an environment over which you have zero control.

That's not to say that client side validation is useless. Quite to the contrary, the client side is where the user experience can be enhanced tremendously through the use of validation, feedback, automatic calculation, Ajax and the like. This is where jQuery has really shined and emerged to become the de facto standard we all use. Can't figure out how to organize a sortable table? Don't sweat it, jQuery has a plugin for that. Need to insure that an email is valid for format? No problem, jQuery has a function just for that purpose. Want to send out telepathic vibes into space calling for the invasion of the Garbonzian horde? I'm not sure but I think jQuery can help with that - try googling for the jQuery megalomaniacal telepathy plugin.

Meanwhile, don't you dare start thinking that all this validation and UI hoopla has anything to do with security. It does not make you more secure and indeed may lull you into thinking you are more secure than you actually are? Some of the JS validation actually keeps things like scanning engines and PCI compliance tools from throwing up red flags. Sometimes automated parsers can't effectively parse through the JS functions in your code to find a way to POST your data and even *check* for vulnerabilities. So you may pass an automated scan test but in fact still be vulnerable. I hope that keeps a few sys admins up at night - it should. Every month I am engaged in reviewing code for applications where security is a must - and I'm finding more and more vulnerabilities that are simply not obvious to automation but *are easily detectable by anyone who knows where to look*.

You don't believe me? How about a real world example I found on an ecommerce site? Consider the following 3 form elements:

```
Quantity: <input type="text" name="qty" value="5" id="qty"/>
<input type="hidden" value="11.40" name="taxes" id="tx"/>
<input type="hidden" value="156.35" name="grandTotal" id="gtot"/>
```

This developer was thorough. Ajax would send the quantity and SKU to the server where it would return Json values for amount, taxes and grand total. The hidden fields collected this information and some HTML was updated to display it to the user. Upon submission an *additional* Ajax request was made to "double check" the amount against the server values again. This second check was to prevent anyone from tampering with the values (presumably) or updating the quantity without rerunning the "update" function. If this last check is successfully performed the jQuery client submitted the values to the server for processing via a POST request.

Ok Muse readers, does anyone see the fatal flaw in this way of doing things? The developer was clearly double checking everything right? By the time he submitted he (or she) knew for sure the values were correct. He is even using the server for these validation checks right? Actually if I simply disable the final post, change the values of

the quantity and grand total and post directly to the handler I can (or I could until I fixed this issue) buy something in any quantity for any amount I wished. The final handler code actually accepted the values, charged the CC and put the values into the DB as is. Think about it. I can purchase 500.00 worth of items, then change the amount to 1.00. Talk about a discount plan! I guess the real question is, why go through all that server validation trouble with Ajax and then fail to run the same CFC functions on the server prior to the final disposition of the data. Maybe that was on his to do list.

The Take Away

So what do we surmise from this cautionary tale? First, the server side is where all validation should take place. Everything else is just eye candy. Important eye candy to be sure, but server side validation is the foundation of your security on any user form. Secondly, server side validation should take place in a certain order. It should always take place immediately before handling - before the final disposition - of the variables submitted. And finally, server side validation must take place in the same HTTP request as the code handling the values. You cannot validate via the server on request A, then submit the values you have validated on request B. Both the server side validation and the handling of the actual form variables need to be sequential within the same request so that no interference by client side tampering is possible.

Final Thoughts

If you are tempted to see this post as an indictment of jQuery, UI design, interactivity or anything else "client side" please think again. These are all important things that should be addressed. Indeed, when orienting toward the user or customer they become primary items that should be carefully considered and programmed. I'm only arguing that such things as client-side form validation belong in a different discussion than security. Security - at least with regard to handling user input - happens only on the server. The fact is that you simply have no control over the client, period. So remember when you are tempted to think you have improved security by adding client validation you still have some work to do :). As always I welcome comments - but please add to our discussion and be civil and respectful. Thanks as always for reading.