Does CFArgument Typing Protect Against SQL Injection

Posted At : February 21, 2008 6:17 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Security

This question was asked on one of the several lists to which I subscribe. The author wanted to know if he needed to *do anything else* as long as he was specifying the "type" attribute of the Cfargument tag - or was that sufficient protection against the dreaded SQL Injection Attack (see my previous post on Application Security). Like the Elves of the Shire my answer is both yea and nay. Consider this example:

```
<cffunction name="getMyUser>

<cfargument type="numeric" name="userid"/>

<cfquery name="get" datasource="mydsn">
SELECT *
FROM Users
WHERE userid = #userid#
</cfquery>

<cfreturn get/>

</cffunction>


<cfset user = getMyUser(form.userid)/>
```

In this example, if a clever scripter passed in something designed to inject into the query, say.... ***15 OR userid 0*** ... the function would throw an error because whatever was in form.userid was not numeric. So in this case - yes, the argument scope and the strong (or strongish) typing is protecting you against injection. But now consider this example:

```
<cffunction name="getUsers>

<cfargument type="string" name="userid"/>

<cfquery name="get" datasource="mydsn">
SELECT *
FROM Users
WHERE userid IN (#userid#)
</cfquery>

<cfreturn get/>

</cffunction>
<!--- form.users is a list of IDS--->
<cfset user = getUsers(form.users)/>
```

In this example the site is expecting a list of userids. The argument scope treats that as a string. Consider what would happen if our budding bill gates passed in ***14,15,16) OR userid NOT IN (0***. The resultant where clause would end up being *WHERE userid IN (14,15,16) OR userid NOT IN (0)*. This would not throw an error because it is still a string and allowed by the argument scope.

Conclusion

My take is the same as always. I can think of no reason *not* to use CFQUERYPARAM. It binds the type to the variable in query during the query preparation. Now, in CF8, you can even use binding with caching - taking away the last nettlesome obstacle.