

## Re: how to code to avoid SQL insertion attacks

**Source:** <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-02/2691.html>

---

**From:** Kevin McMurtrie ([mcmurtri\\_at\\_dslextrreme.com](mailto:mcmurtri_at_dslextrreme.com))

**Date:** 02/22/05

Date: Mon, 21 Feb 2005 16:01:26 -0800

In article <0001HW.BE4014DF000824EAF03055B0@news.newsguy.com>, steve <me@me.com> wrote:

[snip]

>

> *and then there is your reply. ( use prepared statements && ?)*

>

> *consider the code:*

>

> *String sql =*

> *"Select object\_code,client\_file\_name,filedatestamp from*

> *client\_code\_java where deleted=0 and rep\_index=?";*

>

> *PreparedStatement st = dbconn.prepareStatement(sql);*

> *st.setString(1, indexkey); // Bind the replication index*

>

> *rset = st.executeQuery(); // Execute Query*

>

>

> *this satisfies your reply of "using ?" and prepared statements*

> *Is it secure?*

>

>

> *hmm*

> *1. String sql = "Select \* from client\_code\_java where ?=null";*

I assume you meant "rep\_index=null".

At worst that casts both null and the column to an assumed type then performs the comparison. It could be a slow index scan but nothing has been injected.

> *2. String sql = "Select \* from client\_code\_java where ?<>null";*

> *( null can be anything and 'nothing', it is null)*

Same deal.

> *3. indexkey=""*;

That won't compile, but what you're implying doesn't inject anything. PreparedStatement parameter values are safe, even capable of taking binary strings. Read the JDBC specification.

> *so far that is 3 ways to attack this "perfect" system.*

Two ways to maybe cause a full index scan but no unauthorized data has been returned or added. In other words, no injection attack.

I hate superstitious coders. Rather than read specifications and test code in a source-level debugger, they invent invalid facts to support limited observations. Their code becomes a mess of superstitious hacks that performs poorly, is as brittle as glass, and can't be read by anyone.

> *that's not even considering how to easily recover passwords from an oracle  
> thin JDBC connection.  
> Which would make any prepared statement and '?' F\*\*&K useless.*

You're full of it. Getting the password from a JDBC connection is likely possible but it can't be hacked out with a special PreparedStatement bind value.

>  
> *now consider how you would hack:*  
>  
> *String The\_qry = "{ call  
> external\_user.fgfdgfdgfg.asa(?,?,?,?,?,?,?,?,?,?)}";*  
>  
> *where "external\_user" has connect privs. only.*  
>  
> *replies on a postage stamp please.*  
>  
> *if people require help, then help, but if you have not thought about the  
> question then STFU.*  
>  
> *Steve.*

Good concluding advice.