

Re: how to code to avoid SQL insertion attacks

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-02/2690.html>

From: Lee Fesperman (*firstsql_at_ix.netcom.com*)

Date: 02/22/05

Date: Mon, 21 Feb 2005 23:56:33 GMT

steve wrote:

>

> *On Sat, 19 Feb 2005 18:00:34 +0800, Lee Fesperman wrote*

> *(in article <42170CAE.5300@ix.netcom.com>):*

>

> > *steve wrote:*

> > >

> > > *On Wed, 16 Feb 2005 06:58:52 +0800, Lee Fesperman wrote*

> > > *(in article <42127D1C.6C@ix.netcom.com>):*

> > >

> > > > *bighead4694@hotmail.com wrote:*

> > > > >

> > > > > *My backend DBMS is Oracle.*

> > > >

> > > > *Great! AFAIK, they have native support for prepared statements.*

> > > > *Simply pass all arguments received externally as ? parameters*

> > > > *to your prepared statement. This will*

> > > > *protect you against SQL insertion/injection attacks.*

> > > >

> > > > *Please respond if you need more help/info...*

> > >

> > > *no it will not!!.*

> >

> > *Kindly explain, or apologize for calling people idiots.*

>

> *Nope , i don't think so, get over it.*

My, my, you do like bluster. However, you did respond to my request for an explanation. Let's see if you can back up that tough talk...

> *Then take a look at the replies.*

>

> *I find it offensive when i see piss poor replies to people who require*

> *genuine help.*

>

> *We have a guy that has requested help, which is fine.*

> *However i see only 1 reasonable reply, and that is to use stored procedures,*

> *and call outs. (remembering that the*

comp.lang.java.programmer: Re: how to code to avoid SQL insertion attacks

- > *'help' did not give any information on the application/security level, but*
- > *DID state he was using Oracle)*

He only mentioned Oracle after I queried him about it.

- > *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*

Agreed, that looks good to me.

- > *Is it secure?*
- >
- > *hmm*
- > *1. String sql = "Select * from client_code_java where ?=null";*
- > *2. String sql = "Select * from client_code_java where ?<>null";*
- > *(null can be anything and 'nothing', it is null)*

That has nothing to do with the code you posted above. But, anyway...

I don't know that non-standard syntax (does Oracle support that?) I can guess at two meanings:

- 1) It works like `?=b`, `?<>b` where `b` is `NULL`. In SQL, either one would produce `UNKNOWN` (treated as `FALSE`). So, no problem here.
- 2) It works like (or you really meant) `? IS NULL` or `? IS NOT NULL`. Since `?` can't be `NULL` (except by egregious programmer error ... more below), only `? IS NOT NULL` would be a problem. However, that is also an egregious programmer error which just as well could happen in the stored procedure. Yeah, I know that stored procedures are often more carefully crafted, but that is no guarantee. I notice you also didn't advise against stupid mistakes.

If neither conjecture is right, please explain the meaning of the constructs.

- > *3. indexkey=""*;

This is mangled Java and will not compile. I have to guess again as to your meaning.

Are you saying that placing a single/double quote in the value passed for a ? parameter will cause Oracle to misbehave? I am surprised. Either Oracle doesn't provide 'native' support for prepared statements, or it is badly broken. Which is it, and how does a stored procedure avoid the same problem?

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

So far, your grade is 'poor'. Note also that I said nothing about a perfect system.

> *that's not even considering how to easily recover passwords from an oracle*
> *thin JDBC connection.*

I'm not aware of that horrible flaw. What does that have to do with prepared statements?

Oh, I see, you're talking about privileges. I'll touch on that below.

> *Which would make any prepared statement and '?' F**&K useless.*
>
> *now consider how you would hack:*
>
> *String The_qry="{ call*
> *external_user.fgfdgfdgfg.asa(?,?,?,?,,?,?,,?,?,,?)}";*
>
> *where "external_user" has connect privs. only.*

Obviously, the external user has privileges to call stored procedures. Thus, your solution is to simply obfuscate names. You could do the same to table names.

This is not to say I am opposed to stored procedures. They do fill in some very obvious weaknesses in SQL. They can contain processing that doesn't belong in the client, but those issues were not mentioned by the OP.

Stored procedures have their disadvantages. They are often proprietary and obtuse. They require knowledge of yet another language besides Java and SQL. That can introduce more coding errors and decimate any hope for portability.

```
--  
Lee Fesperman, FFE Software, Inc. (http://www.firstsql.com)  
=====  
* The Ultimate DBMS is here!  
* FirstSQL/J Object/Relational DBMS (http://www.firstsql.com)
```