

Re: SQL Injection Prevention

Source: <http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.security/2004-09/0242.html>

From: Nigel Rivett (sqlnr_at_hotmail.com)

Date: 09/28/04

Date: Tue, 28 Sep 2004 11:29:09 -0700

I was pointing that you're not considering it in isolation but with the relative competencies of the developer.

1,2 Yeah that was a bit of a convoluted counter-example – but as I say I've seen it happen.

But if the user doesn't have permissions on the tables (as they shouldn't) then the dynamic sql won't work so there's no problem and possibly the most secure solution.

I think it's much more likely that an application developer would build the sql string from input that the database developer would do it in dynamic sql. As I say it's easy to prevent in a stored proc – is it easy to prevent in the application?

I do it by saying that "you will interact with the database using these objects" but I've never tried to prevent developers circumventing this. I suppose they don't have access to the connection string which stops them.

"Valery Pryamikov" wrote:

- > Nigel,
- > *It's great to disagree on that :-). If you can't consider SQL injection vulnerability in isolation from other factors I can't argue to that. It's just that when you are trying to assess possible security vulnerabilities it's often required to look at possible problems in isolation from others for not being drowned in overcomplexities.*
- >
- > *But frankly, which one do you think is more likely to happen:*
- > *1. inexperienced database programmer putting dynamic sql statement in stored procedure in attempt to quickly solve one or another problem.*
- > *2. database programmer putting dynamic sql statement in trigger procedure.*
- >
- > *Note that in second case it would probably be a bit more experienced database programmer since its more natural to begin with stored procedures than with triggers, but even this could be ignored (I'm not even sure that 2 is even allowed, and if yes, than not on all databases, but again I might be wrong here).*
- >

microsoft.public.dotnet.security: Re: SQL Injection Prevention

> *I'm saying that 1 is much more likely to happen... and if you agree that 1*
> *is more likely to happen than 2, than we have nothing to argue about, and*
> *that's just prove my points. If not – than probably we simply should agree*
> *to disagree.*
>
> *Valery Pryamikov. MCP, MCAD, MCSD, Windows Security MVP (former Windows SDK*
> *MVP).*
> <http://www.harper.no/valery>
>
>
> *"Nigel Rivett" <sqlnr@hotmail.com> wrote in message*
> *news:D6EDE848-C5C2-4C1B-A25F-8D3C40D9507A@microsoft.com...*
> *>>> if we consider SQL injection vulnerability in isolation from any other*
> *> factor*
> *> Don't agree with that.*
> *>>*
> *>> If we include another factor*
> *>>> good contractor bad DB owner situation*
> *>> (well actually good application developer bad database developer)*
> *>>*
> *>> Then you probably wouldn't use stored procedures but do everything from*
> *>> the*
> *>> application – in which case parameterised queries are maybe a good idea*
> *>> but I*
> *>> suspect you would lose a lot of performance having to run any batch jobs*
> *>> from*
> *>> an app (not too different from a situation I'm having to correct now).*
> *>>*
> *>> To continue in the same vein. How do you stop the bad database developer*
> *>> putting a trigger on a table which uses dynamic sql and executes a string*
> *>> including some data just added to the table which comes from user input*
> *>> (I've*
> *>> seen it happen).*
> *>>*
> *>> Similar to the argument about which is better Oracle or sql server? I*
> *>> always*
> *>> say it depends on your expertise*
> *>> Good oracle better than bad sql server and vice-versa.*
> *>> I used to think bad oracle was better than bad sql server – I still think*
> *>> so*
> *>> but sql server is getting more forgiving.*
> *>>*
> *>>*
> *>>*
> *>>*
> *>>*
> *>> "Valery Pryamikov" wrote:*
> *>>*
> *>>> Nigel,*
> *>>> in my other answer to you I mentioned hypothetical "good contractor bad*
> *>>> DB*

> >> *owner situation", that hopefully should better explain my points. Main*
> >> *point*
> >> *is that if we consider SQL injection vulnerability in isolation from any*
> >> *other factor, then parameterized SQL DML actually provides better*
> >> *protection*
> >> *against SQL injection than parameterized call to stored procedure. That's*
> >> *it.*
> >>
> >> *-Valery.*
> >> <http://www.harper.no/valery>
> >>
> >>
> >> *"Nigel Rivett" <sqlnr@hotmail.com> wrote in message*
> >> *news:0317B792-78D8-4FCF-957C-338C2C272F81@microsoft.com...*
> >> > *It's just that your arguement doesn't seem to make sense.*
> >> > *If you write a bad SP it's possible to introduce injection*
> >> > *vulnerabilities.*
> >> > *If you write bad app code it's possible to introduce injection*
> >> > *vulnerabilities.*
> >> >
> >> > *SP's mean that you don't need to give the user permissions on the*
> >> > *tables*
> >> > *(which incidentally means that dynamic sql is not possible so negates*
> >> > *one*
> >> > *of*
> >> > *your arguments).*
> >> >
> >> > *You could say that parameterised client statement security depends on*
> >> > *the*
> >> > *way the client code is written.*
> >> >
> >> > *if instead of the code you gave they decide to exec the statement*
> >> > *concatenated with the parameter they will get back the same cursor but*
> >> > *introduce an injection vulnerability.*
> >> >
> >> > *Basically it's all about writing sensible code – you can't guard*
> >> > *against*
> >> > *poor developers. Trying to force them to use a standard interface will*
> >> > *help*
> >> > *though. A dal which enforces parametrised queries would do that. The*
> >> > *same*
> >> > *as*
> >> > *would only giving access to stored procedures. I suspect they would*
> >> > *have*
> >> > *the*
> >> > *same effect in this area – just that SPs are easier to troubleshoot,*
> >> > *you*
> >> > *can*
> >> > *change the database structure without affecting the app and enhance*
> >> > *security.*
> >> >

> > >
> > > *"Valery Pryamikov" wrote:*
> > >
> > > *Common guys, what is it with you? I'm not bashing stored procedures,*
> > > *not*
> > > *at*
> > > *all. I'm just saying that when it concerns to SQL injection, then*
> > > *parameterized DML statement is more protected than parameterized call*
> > > *to*
> > > *stored procedure. That's it. I don't think you can prove opposite. But*
> > > *that*
> > > *doesn't mean anything about good programming practices what so ever.*
> > > *Do*
> > > *you*
> > > *read subject – SQL injection which only happens due to bad programming*
> > > *practices.*
> > >
> > > *I'm not in any way going to fight beaten to death holy war about "are*
> > > *stored*
> > > *procedures better than SQL DMLs or not".*
> > >
> > > *–Valery.*
> > > *<http://www.harper.no/valery>*
> > >
> > > *"Nigel Rivett" <sqlnr@hotmail.com> wrote in message*
> > > *news:483195D1–E0AC–4765–8EDA–D0B76D923DED@microsoft.com...*
> > > > *You're comparing a well built parameterized sql statement against a*
> > > > *badly*
> > > > *built stored procedure so it's obvious which will win.*
> > > >
> > > > *Stored procedures are easier to review and so catch bad proctices*
> > > > *and*
> > > > *are*
> > > > *usually the domain of people who have some experience in dealing*
> > > > *with*
> > > > *databases.*
> > > >
> > > >> *for stored procedure to return the same cursor as select, this*
> > > >> *stored*
> > > >> *procedure has to execute the same select.*
> > > >
> > > > *Not true.*
> > > >
> > > >> *if stored procedure implemented wrong way – ie it constructs sql*
> > > >> *by*
> > > >> *concatenating received parameter with sql string,*
> > > >
> > > > *I can't believe anyone would do that – if you would consider it then*
> > > > *I*
> > > > *suggest you stay away from databases altogether :).*
> > > >
> > > >

> > > > *The stored procedure in your example would probably be*
> > > >
> > > > *create proc a*
> > > > *@key int*
> > > > *as*
> > > > *select somevalue from sometable where somekey = @key*
> > > > *go*
> > > >
> > > > *You need to build a case that this is more vulnerable than the*
> > > > *apllication*
> > > > *code – bearing in mind that a person writing a stored proc is likely*
> > > > *to*
> > > > *have*
> > > > *more database experience than the person writing the app code.*
> > > >
> > > >>> *in Oracle you have possibility to execute dynamic cursor from*
> > > >>> *stored*
> > > >>> *procedure. Ie. you construct whatever sql string inside stored*
> > > >>> *procedure*
> > > >>> *and open cursor on*
> > > > *that string. I believe it must be similar functionality in SQL*
> > > > *server*
> > > >
> > > > *Your belief is very wrong (and I hope you aren't trying to use that*
> > > > *belief).*
> > > >
> > > > *p.s. I've have never written an explicit cursor in t-sql (except to*
> > > > *"help"*
> > > > *others and never will).*
> > > >
> > > >
> > > > *"Valery Pryamikov" wrote:*
> > > >
> > > >> *Tibor,*
> > > >> *we aren't talking about good programming practices when we discuss*
> > > >> *SQL*
> > > >> *injection, aren't we :-).*
> > > >> *as long as there is possibility to screw something, we have to*
> > > >> *account*
> > > >> *for*
> > > >> *it. Therefore my statement stays that parameterized SQL actually*
> > > >> *provides*
> > > >> *better protection against SQL injection than parameterized call to*
> > > >> *stored*
> > > >> *procedure.*
> > > >>>
> > > >>> *-Valery.*
> > > >>> *<http://www.harper.no/valery>*
> > > >>>
> > > >>
> > > >>

microsoft.public.dotnet.security: Re: SQL Injection Prevention

> >> >>
> >>
> >>
> >>
> >>
>
>
>