

Re: EXEC master..xp_cmdshell Prevention

Source: <http://www.derkeiler.com/Newsgroups/microsoft.public.sqlserver.security/2002-10/3155.html>

From: Kimberly L. Tripp (Kimberly@nospam.sqlskills.com)

Date: 10/24/02

From: "Kimberly L. Tripp" <Kimberly@nospam.sqlskills.com>

Date: Wed, 23 Oct 2002 22:31:39 -0700

On a related note – consider looking at the QUOTENAME function. It's a GREAT way to significantly reduce vulnerabilities around dynamic string execution. If parameters are ALWAYS wrapped in QUOTENAME then hackers will not be able to "inject" problematic code into the database....

For example – the following procedure WILL NOT allow anything but a name to be put into the procedure...

```
CREATE PROCEDURE dbo.CreateDBProc
(
    @DBName sysname
)
AS
DECLARE @ExecStr nvarchar(2000)
SELECT @DBName = QUOTENAME(@DBName, ']')
SELECT @ExecStr = 'CREATE DATABASE ' + @DBName -- + all of the other
stuff to place the files, etc...
SELECT @ExecStr
--EXEC(@ExecStr)
go
EXEC dbo.CreateDBProc 'This is the most stupid :) name I can think of
with [brackets] and all sorts of junk! in the name'
go
```

As opposed to what is often used:

```
CREATE PROCEDURE dbo.CreateDBProc2
(
    @DBName sysname
)
AS
DECLARE @ExecStr nvarchar(2000)
SELECT @ExecStr = 'CREATE DATABASE ' + @DBName --- + all of the other
stuff to place the files, etc...
SELECT @ExecStr
--EXEC(@ExecStr)
```

```
go
EXEC dbo.CreateDBProc2 'this is my test database name'
EXEC dbo.CreateDBProc2 '[fakedbname] DROP DATABASE foo ---'
EXEC dbo.CreateDBProc2 '[fakedbname] EXEC("CREATE PROC Test AS SELECT *
FROM pubs.dbo.authors") DROP DATABASE foo ---'
go
```

OR

```
CREATE PROCEDURE dbo.CreateDBProc3
(
  @DBName sysname
)
AS
DECLARE @ExecStr nvarchar(2000)
SELECT @ExecStr = 'CREATE DATABASE [' + @DBName + '] --- + all of the
other stuff to place the files, etc...
SELECT @ExecStr
go
EXEC dbo.CreateDBProc3 'fakedbname] DROP DATABASE foo---'
Go
```

Anyway, hope this helps!

kt

"Robert Robbins" <rrobbins@kolbnetworks.com> wrote in message
news:urdaqrndpd7096@corp.supernews.com...

> *Hello Beth,*

>

> *I found that my web application did use a connection string in a file
> other than the global.asa. I did not develop this application so creating
> stored procedures for every SQL statement would be impractical. They just
> released a critical update for SQL Injection vulnerabilities but it is
just*

> *a function that replaces single quotes with two single quotes (not to
> mention it introduces unrelated bugs). I have created a new login account
> for applications and an application role for my database. Creating
> application roles is not well documented so I will have to experiment with
> this to ensure I got it right.*

>

> *Robert Robbins*

> *Kolb Net Works*

>

>

> *"Beth Breidenbach" <bbreidenbach@mindspring.com> wrote in message
> news:ehyEhoheCHA.1572@tkmsftngp08...*

> *> Robert,*

> >

> *> I take it you're using string concatenation to build your sql
statements.*

> *I*

microsoft.public.sqlserver.security: Re: EXEC master..xp_cmdshell Prevention

> > *_strongly_ recommend against this as it opens up a whole slew of*
> > *vulnerabilities, including the xp_cmdshell one you're asking about. For*
> > *example, since your login is a dbo a user could: drop your database,*
alter
> > *your table structures, modify your stored procedures, etc..... Far far*
> *far*
> > *better to use parameterized stored procedures and to pass those*
parameters
> > *in via the ado Command object. And of course, once inside the stored*
> > *procedure you still shouldn't be doing string concatenation to build a*
> *query*
> > *(which would just move the vulnerability from outside the sproc to*
inside
> > *it).*
> >
> > *Now, for some housekeeping issues. Let's assume someone figures out how*
> *to*
> > *compromise your database and get sa privileges --> xp_cmdshell runs*
under
> > *the privileges granted to your MSSQLServer service. If I were a betting*
> > *person, I could probably earn \$\$ wagering that people's services were*
> > *running as either the SYSTEM account or a domain admin account -- both*
of
> > *which are way too dangerous. Your SQLServer service requires no more*
than
> > *domain guest privileges. (It often requires _local_ admin privileges,*
but
> > *not _domain_.) This change prevents xp_cmdshell from being used to take*
> > *your LAN.*
> >
> > *Next item -- using dbo for a web application's login is dangerous, for*
the
> > *reasons mentioned in the first paragraph. Many developers use dbo for*
> *their*
> > *app because it makes their development-time work easier. You should*
have
> > *two logins -- one with developer-level privileges (dbo) and one for*
lower
> > *runtime application privileges. Change the connection string for the*
> > *published version (after you've tested it locally).*
> >
> > *As to your original question -- If I create a dbo user and attempt to*
exec
> > *xp_cmdshell via Query Analyzer I receive permission denied errors. It*
> > *sounds as though either a) your connection string isn't what you think*
it
> *is*
> > *(sometimes happens if one or two random asp pages have their connection*
> > *strings hard-coded while the rest of the app shares a single connection*
> > *string) or b) your permissions didn't set the way you think they did.*
> >

microsoft.public.sqlserver.security: Re: EXEC master..xp_cmdshell Prevention

> > *Fire up Query Analyzer, connect as your SQL login user and attempt to exec*
> > *the spoc. What happens? If you receive an error in Query Analyzer but*
> > *not*
> > *in your app, then your app's connection string is goofed up somewhere.*
> *Use*
> > *SQL Profiler to capture your application's request and see what login*
> *the*
> > *page used for its connection. If the spoc works in Query Analyzer,*
> *then*
> > *you need to look more closely at your permissions. Let us know how this*
> > *part turns out and we can try to help you further.*
> >
> > *Best of luck,*
> >
> > *Beth*
> >
> > *"Robert Robbins" <rrobbins@kolbnetworks.com> wrote in message*
> > *news:urao9elubq5024@corp.supernews.com...*
> > > *Hello SQL Server Experts,*
> > >
> > > *I am trying to prevent the SQL Injection vulnerability which allows*
> > *anyone*
> > > *to execute system commands using the xp_cmdshell extended stored*
> > *procedure.*
> > > *My ASP web application uses a connection string with the username and*
> > > *password of a new login account I created. The database has just one*
> > *user,*
> > *my*
> > > *new login account, as the database owner. I've made sure the login*
> > *account*
> > > *has no Server Roles and does not have Master database access.*
> > > *Unfortunately, I am still able to execute that extended stored*
> > *procedure*
> > > *simply by pasting the following code in a text box field used in an*
> > *UPDATE*
> > > *SQL statement. 'EXEC master..xp_cmdshell '[some system command]'*
> > > *Nothing I've tried is preventing me from executing that extended*
> > *stored*
> > > *procedure or any other dangerous stored procedure. Do I need to create*
> > *an*
> > > *application role to use in my connection string? Is it the guest user*
> > *in*
> > *the*
> > > *master database that is providing this permission? Only the sa login*
> > *account*
> > > *has execute permission for this extended stored procedure.*
> > >
> > > *Robert Robbins*
> > > *Kolb Net Works*
> > >

microsoft.public.sqlserver.security: Re: EXEC master..xp_cmdshell Prevention

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