

# [NT] Microsoft SQL Server: Buffer Overflows in numerous extended stored procedures

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-03/0076.html>

---

*From:* [support@securiteam.com](mailto:support@securiteam.com)

*Date:* 03/16/02

From: [support@securiteam.com](mailto:support@securiteam.com)

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: Sat, 16 Mar 2002 12:38:52 +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.

-----

Microsoft SQL Server: Buffer Overflows in numerous extended stored procedures

---

## SUMMARY

Microsoft SQL Server provides the ability to call functions in DLLs outside of the database. These functions, called extended stored procedures, greatly expand the functionality of Microsoft SQL Server. They can be used to access the operating system or the network. Several hundred are shipped with Microsoft SQL Server and custom developed extended stored procedures can be added to the database by the administrator.

Numerous extended stored procedures in Microsoft SQL Server 7 and 2000 contain buffer overflows. These buffer overflows result in several scenarios:

- 1 – The database server crashing
- 2 – The memory in the stack being overwritten, including the return address of the calling function resulting in an exception being thrown

These buffer overflows are the result of passing unusually long strings of data as parameters to extended stored procedures. The majority of these buffer overflows are the result of passing large Unicode buffers. Although these buffer overflows are based on Unicode strings, exploiting them is not particularly difficult given such methods as the "Venetian" exploit which allows arbitrary shell code to be written using Unicode buffers.

## Securiteam: [NT] Microsoft SQL Server: Buffer Overflows in numerous extended stored procedures

### DETAILS

Vulnerable systems:

Microsoft SQL Server 7

Microsoft SQL Server 2000

The following extended stored procedures are vulnerable:

```
xp_controlqueueservice
xp_createprivatequeue
xp_createqueue
xp_decodequeueecmd
xp_deleteprivatequeue
xp_deletequeue
xp_displayqueueemesgs
xp_dsninfo
xp_mergelineages
xp_oledbinfo
xp_proxiedmetadata
xp_readpkfromqueue
xp_readpkfromvarbin
xp_repl_encrypt
xp_resetqueue
xp_sqlinventory
xp_unpackcab
```

Based on the findings of Cesar Cerrudo ([sqlsec@yahoo.com](mailto:sqlsec@yahoo.com)), the extended stored procedures listed above contain buffer overflows that allow arbitrary shell code to be inserted onto the stack.

By default, a few of these extended stored procedures are granted to the public group. Those that are granted to the public group allow a non-privileged user to gain full control of the operating system and the database. For those extended stored procedures that are not granted to public by default, several concerns still exist:

- 1) The DBA for one instance can gain full control of the operating system and other instances and programs on the server.
- 2) If a DBA has granted permissions to execute one of these extended stored procedures to a non-privileged user, that user can gain full control of the system.

The problem is caused by assumptions in the extended stored procedures about the data types and sizes being passed in. Extended stored procedures do not accept parameters directly as other functions do. Instead, an extended stored procedure retrieves the parameters that are passed to it using a set of API functions, the most important of which is `svr_paraminfo`. Below is the definition of this function:

```
int srv_paraminfo (
SRV_PROC * srvproc,
```

## Securiteam: [NT] Microsoft SQL Server: Buffer Overflows in numerous extended stored procedures

```
int n,  
BYTE * pbType,  
ULONG * pcbMaxLen,  
ULONG * pcbActualLen,  
BYTE * pbData,  
BOOL * pfNull );
```

A stored procedure checks to determine the data type and length of each parameter passed to it. A varchar is limited to 8000 characters and an nvarchar is limited to 4000 characters. If however an ntext data type which has a maximum length of  $2^{30} - 1$  (1,073,741,823) characters is passed to a parameter which expects a nvarchar and the extended stored procedure attempts to copy this parameter into a buffer created for a much smaller nvarchar, the stack can be overwritten.

Further research of the internal buffer overflows done by the SHATTER team is available at:

<<http://www.appsecinc.com/resources/alerts/mssql/02-0000.html>>  
<http://www.appsecinc.com/resources/alerts/mssql/02-0000.html>

### ADDITIONAL INFORMATION

These vulnerabilities were discovered by <<mailto:sqlsec@yahoo.com>> Cesar Cerrudo.

=====

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](mailto:list-unsubscribe@securiteam.com)

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

- 
- **Previous message:** [support@securiteam.com](mailto:support@securiteam.com): "[UNIX] FreeBSD Mod frontpage Port Contains Exploitable Buffer Overflow"
  - **Messages sorted by:** [\[ date \]](#) [\[ thread \]](#) [\[ subject \]](#) [\[ author \]](#) [\[ attachment \]](#)