Global Information Assurance Certification Paper

Microsoft SQL Server 2000 Security Overview

Joe Partlow

July 10, 2002

GSEC option 1, version 1.4

Abstract

In today's increasingly networked world, information storage and retrieval is absolutely vital to the continued success of the Internet. Databases are relied upon more everyday, but with that reliance comes potential disaster by way of customer credit or personal information being stolen, corporate trade secrets being compromised, lost business revenue, or even human lives being endangered by medical records being altered. All of these worst-case scenarios are possible if the information in your database is altered, made unavailable or deleted by accidental or purposeful means. Business demands 100% data integrity at all times, so properly securing your most valued software asset is a must. Microsoft's SQL Server 2000 is many companies' means of information storage and retrieval and provides many different ways to minimize this risk, but only if it has been securely configured and maintained.

With the default installation of any software package, there are potential security risks and vulnerabilities. SQL Server 2000 is no exception, but I will demonstrate how to minimize your exposure and maintain a secure database server with relative ease. This paper will give a brief overview of SQL Server 2000, how to securely install it, how to securely connect to it, and outline some current vulnerabilities and ways to help protect against them. Basic knowledge of SQL Server 2000 is needed to fully understand the severity of these issues. At the end of the presentation you should be equipped with the basic knowledge of how to protect and securely maintain your own SQL Server 2000.

SQL 2000 Introduction

SQL Server 2000 is Microsoft's flagship relational database management system (RDBMS) with thousands of installations worldwide. With it's ease of use, enterprise-level speed, scalability and relatively low Total Cost of Ownership (TCO) over other RDBMS such as Oracle, SQL Server 2000 is gaining popularity and market share quicker than ever and provides many new features and improvements over it's predecessor, SQL Server 7.0 [2]. Some of the major new improvements over 7.0 include XML support, new data mining features included in Analysis Services, automatic support of data and network traffic encryption and meeting the NSA's C2 security classification. Read more about this important classification here [9]:

http://www.radium.ncsc.mil/tpep/epl/entries/TTAP-CSC-EPL-00-001.html

Secure Installation

Before installing SQL Server 2000, make sure your Windows 2000 or NT server is secured. Installing SQL Server on an unsecured Windows Server is just as damaging or worse than not securing SQL Server itself. Securing the Windows 2000 and NT operating system is beyond the scope of this paper, but an excellent starting point is Microsoft's Technet security home [3]:

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/prodtech/windows/windows2000/staysecure/

Another good starting point is the SANS Securing Windows 2000: Step by step guide available at:

http://store.sans.org/store_category.php?category=consguides

The first step to securely install SQL Server 2000 is to make sure all SQL Server system files, data files and logging are installed on a NTFS partition. This allows access control lists (ACLs) to be applied to all files of the installation to minimize the chance of unauthorized access.

After choosing the appropriate partition, the next choice is which authentication method to use when logging into SQL Server: Integrated Windows Authentication or Mixed Mode Authentication. The default installation choice, Windows Authentication, uses Windows to authenticate users and establishes a trusted connection to SQL Server. Logins are tracked via the Windows security identifiers (SIDs) for trusted connections or SQL Server generated GUIDs for non-trusted connections. In Mixed Mode Authentication, SQL Server can use Windows Authentication (if the client and server are both capable of NTLM or Kerberos authentication) or database access can be granted using a built-in SQL Server login and password contained in the **sysxlogins** table in the **master** database [10]. There are pros and cons to each method and each depends the specific business needs, but Windows Authentication is generally the safer method because 'sa' passwords can potentially be stored plaintext in the SQL installation files or in the system tables with weak encryption [10]. If Mixed Mode Authentication is used, be sure to set a strong password for the 'sa' account. If Windows Authentication is used, make sure that the Windows user account is limited to minimal access on the server. However, if the Windows account becomes compromised, other Windows services running on the server may also be compromised instead of just the SQL user's account access.

After choosing the authentication mode, the four SQL Server services (MSSQLServer, SQLServerAgent, Microsoft Search Service and MSSQLServerOLAPService) must be configured to use one of the following

types of Windows accounts: Local User, Local System or Domain User Account. The safest of these accounts is a Local User Account with logging enabled because the account can be setup to only access to what is specifically required by SQL Server and audited for non-SQL access much easier. Local System Accounts are not recommended [4] because they can give administrative access to the server if SQL Server is compromised and Domain User Accounts are not recommended because if the account get compromised, other resources on the network accessible by that domain account could be compromised as well. It is actually better to not have the SQL Server in a domain if accessed by external remote clients or web servers.

After the type of user is chosen, it is imperative that some level of auditing be enabled for the SQL Server user accounts. Auditing can be configured to write all successful or unsuccessful logins to the Windows event log or by using the SQL Profiler to capture all actions performed on SQL Server. SQL Profiler is a powerful new auditing tool that can capture almost all end user activity including SQL commands, server shutdown or restart and most other SQL Server events. The Profiler can capture the information to display real-time and write to a text file or database table for later analysis.

After the SQL installation has been finished, there are some other built-in functions of SQL Server that should be disabled or restricted. The main areas to concentrate on are the removal or restricting of access to some potentially damaging extended stored procedures and setting the appropriate user roles on the server. Many built-in extended or OLE automation stored procedures have had buffer overflows reported in the past and some have powerful maintenance functions that are usually not required in a production environment. According to SQLSecurity.com, some of the most powerful stored procedures that should be locked down or removed are [1]:

Sp_OACreate	**xp_perfend**
Sp_OADestroy	**xp_perfmonitor**
Sp_OAGetErrorInfo	**xp_perfsample**
Sp_OAGetProperty	**xp_perfstart**
Sp_OAMethod	**xp_readerrorlog**
Sp_OASetProperty	**xp_readmail**
Sp_OAStop	**Xp_regaddmultistring**
sp_sdidebug	**Xp_regdeletekey**
xp_availablemedia	**Xp_regdeletevalue**
xp_cmdshell	**Xp_regenumvalues**
xp_deletemail	**Xp_regremovemultistring**
xp_dirtree	**xp_revokelogin**
xp_dropwebtask	**xp_runwebtask**
xp_dsninfo	**xp_schedulersignal**
xp_enumdsn	**xp_sendmail**
xp_enumerrorlogs	**xp_servicecontrol**
xp_enumgroups	**xp_snmp_getstate**
xp_enumqueuedtasks	**xp_snmp_raisetrap**
xp_eventlog	**xp_sprintf**

xp_findnextmsg	**xp_sqlinventory**
xp_fixeddrives	**xp_sqlregister**
xp_getfiledetails	**xp_sqltrace**
xp_getnetname	**xp_sscanf**
xp_grantlogin	**xp_startmail**
xp_logevent	**xp_stopmail**
xp_loginconfig	**xp_subdirs**
xp_logininfo	**xp_unc_to_drive**
xp_makewebtask	**xp_dirtree**
xp_msver	

Setting the user roles and database access is often overlooked when installing SQL Server. A bad habit many sites get into is always using the 'sa' account to remotely connect to the database rather than going through the trouble of setting up users in the correct security roles. Roles are used in SQL Server similar to the way Windows uses groups and can be server-wide (fixed server roles) or defined at the database level (fixed database or user-defined roles). Roles can be nested within another and a user can belong to more than one role at a time. The SQL Server 2000 fixed server roles are defined below [10]:

- **Public** – The public role is required for each database and contains all default permissions and users for each database. It functions similar to the "Everyone" group in Windows.
- **Sysadmin** – All members are able to perform any SQL Server activity. The local administrators of the Windows server are included automatically in this role.
- **ServerAdmin** – This role is able to configure server-wide options and may shut down the server.
- **SetupAdmin** – Members of this role may manage linked servers and startup procedures.
- **SecurityAdmin** – Any member of this role can manage server-wide security settings, linked servers, reset passwords and create databases.
- **ProcessAdmin** – ProcessAdmins can terminate any process running on the SQL Server.
- **DbCreator** – This role may create, alter, drop and restore all databases.
- **DiskAdmin** - This role is permitted to manage all disk files.
- **BulkAdmin** – BulkAdmins are able to run the BULK INSERT command without being members of the **sysadmin** role.

The default SQL Server fixed database roles are:

- **db_owner** – Performs all database maintenance and configurations.
- **db_accessadmin** – Able to add or remove access for Windows users and groups or SQL users.
- **db_datareader** – Can read all user data tables.
- **db_datawriter** – Can add, delete or change all data in user tables.
- **db_ddladmin** – Able to run any data definition language (DDL) command.
- **db_securityadmin** – Manages roles and permissions.
- **db_backupoperator** – Able to backup the database.
- **db_denydatareader** – Explicitly denied read access to the database.
- **db_denydatawriter** – Explicitly denied write access to the database.

Before deploying SQL Server, the server's registry should also be protected. This can be accomplished by removing the permissions for the 'everyone' group and adding permissions for the 'administrators' group on the following keys:

- HKEY_LOCAL_MACHINE \SOFTWARE \Microsoft \MSSQLServer (or instance name)
- HKEY_LOCAL_MACHINE \SOFTWARE \Microsoft \Microsoft SQL Server\InstanceName

Some other functions that should be disabled, if not being used, are the SQL Mail functionality to avoid SMTP vulnerabilities and the SQLXML functionality to avoid current buffer overflows. As with any software application, disable any unnecessary services or options to avoid future problems. Just because something isn't known as vulnerable today, doesn't mean someone isn't keeping the exploit for private use or working on the next one!

Secure Use and Connections

Making secure connections to the SQL Server database is just as important as securing the SQL Server itself. There are many things to look for when securing applications that connect to a back-end SQL database, but some of the key areas are securing remote clients or web servers, validating user input, not using plaintext passwords in database connection strings and connecting on a port other than 1433. I will go into more detail on these topics below.

One of the most important reasons to validate user input residing on web pages is to avoid SQL code injection into the form fields. According to an Information Security magazine definition "SQL injection is the process of extending normal user input to include database queries [11]." SQL commands can be injected into input fields by appending or inserting a single quote and a semicolon(';) in the input field followed by the SQL command you wish to run and two dashes (--) to terminate the existing query without generating an error. For example, in a logon field type the following:

Login: '; drop table users --

This command could cause the users table to be deleted if the SQL Server is not properly protected against these types of attacks. Another example of injection could be carried out by simple trial and error to find out the database structure and entering in a new user by typing in the following:

Login: '; insert into users ('new_username','new_password') --

Additional papers available by NGSSoftware go into much more detail about SQL injection [6].

When connecting to SQL from Microsoft's Internet Information Server (IIS) or any other web server, it is imperative that the web pages be secured in order to

avoid accidentally disclosing embedded usernames or passwords. Securing the web pages against source disclosure can be done by setting the appropriate NTFS permissions on the files and turning of the debugging messages returned by the server in case of an error. In IIS, debug messages can be turned off by going to the Internet Services Manager, properties, home directory, configuration, App debugging, then checking the box "send text error message to client" or they can be turned off under the "Custom Errors" tab and setting a custom web page or URL. Most types of dynamic web pages also use connection strings to access the back-end databases. Unfortunately, most times these connection strings have the username and password to the database written in plain text inside of them. In order to avoid using a hard-coded SQL username and password in connection strings, it is recommended to use windows authentication instead of mixed mode authentication for database connectivity. The connection string would be coded:

```
"Data Source=USER_DSN;Integrated Security=SSPI;Initial Catalog=USER_DB"
```

instead of this:

```
CnString = "DSN=USER_DSN;UID=SQL_USER;PWD=USER_PASSWORD"
Application("strConnect")= CnString
```

SQL Server 2000, by default, will listen on port 1433 for client connections. It is recommended to change this port to something other than 1433 to prevent automated attacks from scripts or worms that are hard-coded to look for this port. While changing the port number will not stop someone from easily finding out the real port by using a tool such as **SQLPing2** (available from **http://www.sqlsecurity.com**), it will help shield against automated attacks that look for this default port. The SQL Server should also be installed in a dedicated DMZ environment with all traffic to and from the server encrypted with the IPSec or SSL encryption protocol. Port 1433 and 1434 should also be filtered or ACLs applied at all border gateways in the DMZ to avoid unauthorized access [3].

Current Vulnerabilities and Exploits

A SQL Server 2000 installation may be open to many damaging vulnerabilities including buffer overflows, password cracking or denial of services. It is very important to scan your network frequently to find unknown or rogue SQL Servers so you aren't caught by surprise with a compromised SQL Server you never knew existed. Many proprietary software packages such as voice mail systems, shipping and manifesting systems and inventory systems use SQL Server as their back-end database, which can be installed on the machine without the user ever knowing it. SQL Servers can easily be identified on the network and enumerated by using a tool such as **SQLPing2** or commercial offerings by Application Security Inc (http://www.appsecinc.com) and NGSSoftware (http://www.nextgenss.com). For example, SQLPing2 will scan an IP address range over UDP 1434 and return the SQL Server name, instance

name, cluster status, version and any network library support details and even brute-force password check [1]. After this information is gathered, attacks can be carried out against the SQL Server or it's vulnerabilities by using the built-in Query Analyzer (or isql/osql) to access the database, packet sniffers to grab unencrypted passwords on the network or other brute-force password crackers like **sqldict** by Arne Vidstrom, or **sqlbf** and **sqlpoke** by xaphan [4]. Some other specific exploits that can be used against SQL Server include:

- Installation log files left on the system with plaintext or poorly encrypted passwords. If the appropriate ACLs for these files have not been applied, the passwords can be retrieved by any user on the system.
- An unchecked buffer in SQLXML, if enabled, could allow an attacker to run code of their choice on the Microsoft IIS Server. It could also allow scripts to run at a higher privilege on the client's computer [7].
- SQL worm 'Spida' attacking SQL Servers with blank 'sa' passwords by enabling the guest account and adding it to the local or domain administrators group and compromising the SAM and other SQL Server settings.
- Buffer overflows in many extended stored procedures and remote data source queries that can allow code to run in the security context of the SQL Server or cause the SQL Server service to fail [7].
- Web server source disclosure or SQL injection techniques explained above that can reveal account passwords or data structures because of poorly designed web pages.
- If the SQL Server Agent service is running as the 'localsystem' user, Dave Litchfield of NGSSoftware has discovered [12] that key operating system files can be overwritten by outputting the results of a job scheduled by a member of the public role, into an existing file. This potentially allows arbitrary code to be executed or the file to become corrupt.
- One of the most creative exploits against SQL Server is described in the excellent paper by Chris Anley of NGSSoftware. In his paper, Chris demonstrates how it is possible to become a sysadmin of SQL Server by catching C++ exceptions generated by SQL Server and reverse engineering the code to change the UID to 1 allowing for sysadmin privileges without detection [6].

If your SQL Server is open to any of the above vulnerabilities, it's probably only a matter of time before it is compromised. In order to prevent this, the following Microsoft patches and updates should be applied [2]:

1. http://www.microsoft.com/sql/downloads/2000/sp2.asp **(Service Pack 2)**
2. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-043.asp **(Cumulative security updates for SP2)**

Patches and updates are a definite, but encrypting SQL traffic between hosts with IPSec filters or SSL Encryption and enforcing strong account passwords

should also be done to protect against password sniffing and brute-force attacks. If the TCP/IP netlib is the only network library being used, it is also possible to "hide" the server from enumeration tools such as SQLPing2 by setting the Server Network Utility 'Hide Server' property or changing the following registry key to 1 [1]:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\SuperSocketNetLib\Tcp\TcpHideFlag

Physical security is also very important. If the servers are kept in an open area they could be subject to brute force operating system attacks, forced shutdown or theft of the server itself. Backup media must also be securely stored to avoid the databases being copied and re-attached to unauthorized servers!

Security Checklist and Summary

Here is a quick checklist to help securely setup and maintain SQL Server 2000:

- Make sure your Windows 2000 or NT Server installation is secure before SQL Server 2000 is installed.
- Install data, logs and system files on NTFS partitions with appropriate ACLs.
- Physically secure the server and all backup media.
- Use Windows Authentication instead of the built-in SQL user accounts whenever possible.
- Make sure the built-in 'sa' and any other SQL accounts have strong passwords.
- Run the SQL Services under a local Windows 2000 or NT account instead of the localsystem account.
- Remove or allow only administrative access to extended stored procedures such as **xp_cmdshell** and **xp_regwrite**.
- Remove the 'guest' user from all databases (except the required master and tempdb).
- Limit interactive and ad-hoc query access to the server to avoid easy escalation of privilege attacks.
- Encrypt traffic between hosts using IPSec or SSL if possible.
- Disable SQL Mail if possible in order to avoid common mail vulnerabilities.
- Block and/or filter port 1433 and 1434 (or alternate SQL ports) at all possible gateways.
- Secure web servers or other clients to avoid a remote compromise or source code disclosure.
- Enable logging and auditing of all SQL service and user accounts.
- Download and install all current hotfixes, patches and service packs for SQL Server as well as the Windows operating system.
- Keep on current vulnerabilities by subscribing to mailing lists by Security Focus, Microsoft and SANS.

SQL Server has many security enhancements built-in and available from third parties to help protect it's data, and with the addition of regular security auditing and maintenance it can provide for a solid, secure database platform for any environment whether it's on a private intranet or an internet-facing network.

References

1. SQLSecurity.com
   http://sqlsecurity.com/DesktopDefault.aspx

2. Microsoft SQL Server 2000 Home
   http://www.microsoft.com/sql/default.asp

3. Product Support Services Informational Alert on SQL Server
   http://www.microsoft.com/security/security_bulletins/ms02020_sql.asp

4. Scambray, Joel and McClure, Stuart. Hacking Windows 2000 Exposed Osborne, 2001
   http://hackingexposed.com/win2k/home.html

5. NGSSoftware: Violating database security mechanisms
   http://www.nextgenss.com/papers/violating_database_security.pdf

6. NGSSoftware: SQL Injection
   http://www.nextgenss.com/papers/advanced_sql_injection.pdf
   http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf

7. Security Focus
   http://online.securityfocus.com

8. Using Windows security with IIS and SQL Server 2000
   http://www.ntsecurity.net/Articles/Index.cfm?ArticleID=23035

9. SQL Server 2000 C2 Administrator's and User's Security Guide
   http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/sql/maintain/security/sqlc2.asp

10. Microsoft Technet Resource Kit – Implementing Security
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/sql/reskit/sql2000/part3/c1061.asp

11. Scoudis, Edward. "Cracker Tools and Techniques." Information Security Magazine July 2002 (2002)

12. NGSSoftware: Arbitrary File Creation/Overwrite with SQL Agent Jobs
    http://www.nextgenss.com/advisories/mssql-jobs2.txt