



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at <http://www.giac.org/registration/gsec>

Securing SQL Server 2000 – A summary of practices

Ernesto F. Rojas

April 5, 2003

GSEC option 1, version 1.4b

© SANS Institute 2003, Author retains full rights.

Introduction

Recently the importance of Microsoft (MS) SQL Server 2000 based database systems came to the world's attention in the form of the Slammer Worm, which infected thousands of servers, bringing down Internet and commercial applications across multiple industries and countries. The patch to stop this malicious code was available from Microsoft for nearly 6 months prior to the attack and yet many System Administrators (Sysadmins) and Database Administrators (DBAs) had not installed the recommended patch, causing the exposure of their systems and applications. Part of the reason claimed for the poor level of upgrades is the complexity of SQL Server administration and patch system. As SQL applications become more prevalent, due to the familiarity of programmers with the SQL language and the development of tools that easily connect to databases, it is imperative that the practice of security administration and the patching of this product becomes easier to use by better documentation.

There is little documentation available that covers all areas related to SQL Server 2000 security topics because of the diversity of disciplines involved, from programming to administration. Recently published however, is a book on this subject of security for SQL 2000, plus an operating guide from Microsoft, but the majority of the documented practices are a quilt of articles in magazines and the Internet. This paper is intended to assist the reader in securing their SQL Server 2000 by documenting the various recommended practices utilized today.

Securing SQL Server

The task of securing SQL Server is not as simple as securing other systems, because of the flexibility Microsoft built into the product. In addition, because SQL Server is the internal core and data repository, there are substantial differences of opinion and misunderstanding between developers and DBAs on how to secure SQL Server. The goals of the DBA or Sysadmins in securing a SQL server system should be: Data Confidentiality, Data Integrity and Data Availability. It is by following these 3 basic goals along with the assistance of the developer community that one can achieve a secure SQL Database. The developer community needs to close the openings created by poor coding practices which have allowed exploits like sql injection and other input screen malicious code. The administrative function needs to apply patches and employ secure practices in operations.

Recently, following the SQL Slammer incident a number of articles have been published restating the most basic steps required in securing SQL Server 2000. For the uninitiated reader it is important that I reiterate these steps, because they are a foundation to good SQL security. However by no means should they be considered the end of the journey or a one step cure to completely secure a database server. The most often noted security recommendations on how to maintain a secure server are:

- Make sure that your server is always up-to-date with the most recent Microsoft Service Packs and patches. It is important to note here, that the people who did not have to spend the weekend after the Slammer Worm appeared shutting down SQL Servers and patching them were the ones that had up-to-date server patches. The fix for Slammer was issued in July 2002, 6 months before the incident took place. There was more than adequate time to test and patch production systems.

There is a simple way for a security analyst to verify the level of the SQL Server updates based on the SQL version number. Every time Microsoft issues a patch or service pack for SQL it increments the version number of the product. There are tables available that show the incremental version numbers and corresponding patches that have to be installed in order to be at that version level. For example, as of the date of this paper, the latest version for SQL 2000 is 8.00.760; this means that Service Pack 3 is installed and for SQL 7.0 it is 7.00.1077. A list I use from frequently is the "SQLSecurity Checklist" from SQLSecurity.com found at:

www.sqlsecurity.com/DesktopDefault.aspx?tabindex=3&tabid=4

It is simple to have SQL Server provide the current version information by issuing the following query to the server:

```
SELECT @@Version
```

This will return the version number as shown in Figure 1:

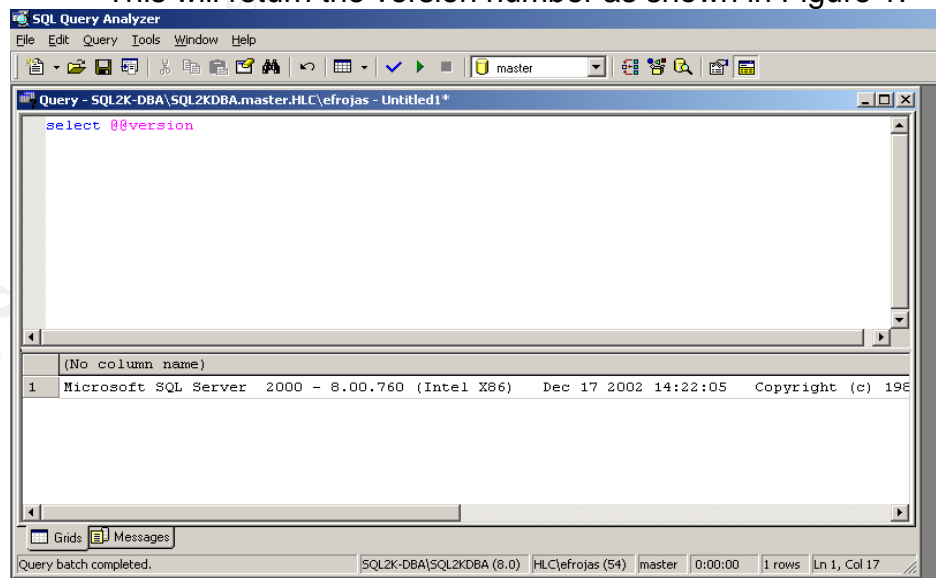
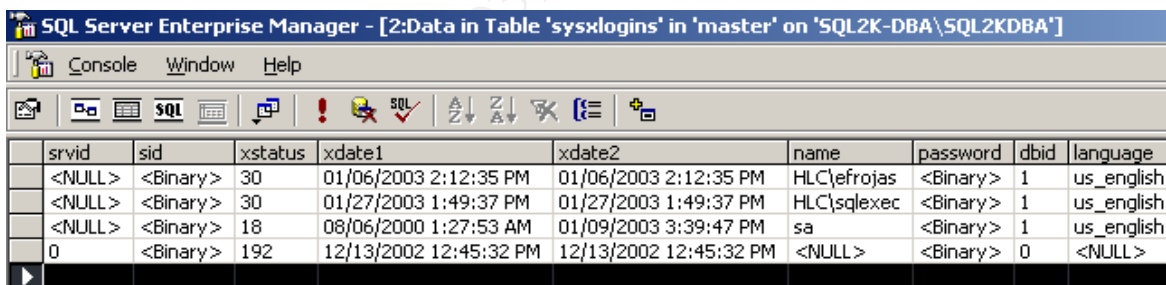


Figure 1

- Do not leave the sa (default administrative account created during the installation of SQL Server) account without a password, even in those cases where you are using Windows Authentication. The reason for this is that it is very simple for someone with remote access to the server registry to change the authentication method on the SQL Server. Unless you have secured the registry by denying remote access to it and limited the access only to designated Database and/or Domain Administrators at the server only, an intruder can remotely access the registry and change from a 1 to a 0 the key located at:

HKLM/Software/Microsoft/Microsoft SQL Server/Instance Name/MSSQLServer/LoginMode

The result of this simple change will modify the SQL Server operating mode from Windows Authentication to Mixed Authentication, exposing the sa user account stored in the master database sysxlogin table to the login prompt in the SQL Server registration property sheet. A sample sysxlogin table of a server running in Windows authentication mode can be seen in Figure 2. Please note in column xdate1 the sa user with a creation date of August 8, 2000 (date of MS production release), as well as the second date in column xdate2 of 9 January 2003, which is the date I entered a complex password for sa to protect against this exploit.



srvid	sid	xstatus	xdate1	xdate2	name	password	dbid	language
<NULL>	<Binary>	30	01/06/2003 2:12:35 PM	01/06/2003 2:12:35 PM	HLC\efrojas	<Binary>	1	us_english
<NULL>	<Binary>	30	01/27/2003 1:49:37 PM	01/27/2003 1:49:37 PM	HLC\sqlservr	<Binary>	1	us_english
<NULL>	<Binary>	18	08/06/2000 1:27:53 AM	01/09/2003 3:39:47 PM	sa	<Binary>	1	us_english
0	<Binary>	192	12/13/2002 12:45:32 PM	12/13/2002 12:45:32 PM	<NULL>	<Binary>	0	<NULL>

Figure 2

In this scenario once you select “Use SQL Server authentication” and provide a login name of “sa” without entering a password you have now registered the server and have full control of it and all its databases. Since no password was set for the sa user, because the sa password was not prompted for during installation in Windows authentication mode, sa is without a password. Once this takes place, the intruder can make any changes to the server in order to meet his desired goals. More about this and other Microsoft recommendations can be found at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/sql_security2000.asp

- To secure the SQL Server's file structure using NTFS permissions on all drives of the server, limit access of the files related to SQL Server to administrators only. There is no reason for regular users of a SQL Server to need access to the server's file structure. SQL Server 2000 does not require for users to have any rights to drives and folders on the server, unless a user application is hosted along with SQL Server on the same hardware. In that case only allow access to the folders necessary for the users to access, in relation to the application. If this is not feasible in your environment, at a minimum, limit access to the locations of the databases, logs and the SQL executable files by restricting access to only Domain and Local Administrators of the Server and to the Domain Account that SQL Server uses to start and run services on the server.

For most organizations that operate SQL Server if they can succeed to implement and maintain the previous 3 bullet points, then they are well ahead in security compared to many of the SQL Servers found today; however there are many more security points with regards to SQL Server 2000 to be covered, especially if your server hosts sensitive or confidential information, which may be misused by unauthorized third parties.

Many users of SQL Server 2000 are not aware that in November of 2000 this software was approved as a C2-compliant application by the DOD, which means SQL Server 2000 meets the requirements of the Department of Defense Trusted Computing guidelines. More can found on this subject at:

http://www.gcn.com/vol19_no29/news/3049-1.html.

A setup guide can be downloaded at:

<http://www.microsoft.com/Downloads/details.aspx?displaylang=en&FamilyID=71C146F3-9907-40CD-BABF-3506ECD33254>

However, unless you are a government contractor or work for one of the 3 letter agencies, it is probable that you do not want to take the effort or spend the resources to secure your SQL Server to the C2 level; therefore the remainder of this article will be spent presenting the most frequently used types of procedures and the means by which they secure SQL Server. Some of these procedures are operational in nature and others are programmatic, but a combination of some or all should make your SQL Server 2000 as a whole more secure than if you only performed the basic three steps mentioned initially.

One can categorize these procedures into 5 areas of activity: Design, Management, Development, Data Security and Auditing. As with every project the more that is addressed in the planning and design stages, the easier the

implementation and operational management is later. I wish to acknowledge at this time that most of this approach comes from, what I consider one of the best text books on SQL Server security, *Managing SQL Server Security* by Mike and Curtis Young, Gearhead Press, 2002 and from various papers written by the Microsoft SQL Team in Redmond, WA, in particular the text from the “pattern & practices” series, *Microsoft SQL Server 2000 Operations* by Microsoft 2002 and other SQL specialists closely related to the subject.

Design

As with any activity a good design provides the opportunity to build into the software the necessary security provisions that later will be used to operate the system and keep the data from falling into the wrong hands. For this article, it is presumed that the reader understands the concepts of operating a design and test environment; but as a caution I want to emphasize that no matter which of these listed steps are taken, they should be developed and tested in a non-production environment, plus adequate evaluation and input from the user community should be obtained before implementing them on a production system. The following steps implemented during the design phase will enhance security during operations.

- Use separate instances (multiple installations on the same hardware) of SQL 2000 to segregate sensitive databases from less sensitive ones. For example, only the DBA and the developer responsible for the system may access a payroll database, while other developers and Domain Administrators may access other databases residing on the same server hardware. Separate instances limit the opportunity for an intruder to gain access to the confidential data.
- Use stored procedures and views to isolate data tables from application users. This allows the designer to deny users permissions to directly read, write, update and delete records from the tables storing the data or the right to modify data directly. Stored procedures can contain parameter field definitions that limit the type of data that a field will accept prior to writing the record. Views limit the type of information that the user sees, based on task requirements or user roles.
- Limit access to batch processes that do not require user input, for example updates, data transfers between systems, data collection from field applications and similar tasks. These should run under the context of the Domain User account under which SQL Server operates or a task account with even lesser permissions. In addition by using the SQL Agent service to run these types of jobs,

they can be scheduled at predetermined times without user or administrator intervention. If there is concern over knowing the status of a job, the SQL mail component can email all concerned parties upon completion and/or failure.

- Examine the requirements for SQL Clustering, log shipping and standby servers based on the application's need for data redundancy and disaster recovery. As the database is designed that is the time to identify any of these operational activities that can be used to maintain high availability. This allows for the design of tables and procedures that accommodate these types of data redundancy techniques.
- Select an account with the least privileges necessary to run SQL Server. This at times is a hotly contested issue amongst and between SQL DBAs and Domain administrators, a great majority of DBA's feel that they should have a Domain Administrator account to discharge their tasks which is necessary, the problem is that in most cases they use the same account to have the SQL services run under for convenience. This approach to using a domain administrator account is contrary to every recommendation from the field today. In order for SQL Server to run multiserver jobs, replication, backups to network drives, join statements involving other databases in other servers and the SQL Server Agent mail facility, then you need to have a Domain User account with special local server rights. See Appendix A from Young 70-71. If there is no need to have the features listed in the previous sentence such as a server in a DMZ, then SQL can be run under a local user account on the server, with even more limited rights. In addition, it is strongly discouraged to run SQL Server under the local system account because if compromised it can give access to the operating system tools, commands and other portions of the server such as the Windows directory and subdirectories.

Management

The management of the server is not only the day-to-day administration of users, databases and backups, but also the initial set up of SQL Server on the hardware along with the configuration and security of the Operating System. SQL Server is very flexible in the number platforms that it supports from Windows 98 to Windows XP on desktops and NT Servers 4.0, 2000 and the soon to be announced 2003. Because of this, it is critical that the System Administrator or DBA have a hand in deciding the selection of hardware and operating system appropriate to support the application within its production operating parameters. In this area there are a number of decisions that can influence server security

and reliability on an ongoing basis, far beyond the initial development and testing of the database application.

- Do not install SQL Server 2000 desktop edition for server based production applications. Besides the limitation of 10 user connections the desktop versions are configured to support development efforts or single user databases. This is a common problem when vendors include the MSDE (desktop edition) as a give away database to store their application data. In many cases the limitation is not discovered until production and 4 -5 power users connect to the server and then performance problems begin immediately. It is at that time that the user community calls the vendor or the DBA for help in solving the problem and finds out that the solution is to install either the Standard or Enterprise Server version, which is licensed based on the number of processors. The primary reason for the give away of the desktop edition is that there is no licensing involved, draw your own conclusions.
- When setting up a production server, see that the server has adequate storage and processor growth capacity to cover a reasonable period of time into the future. In my own experience I forecast capacity for 12 to 18 months into the future so that there is time to study the server demands over a period of 6 to 9 months, to develop an adequate growth curve. It is often that once the user community begins to implement the application, that they discover the need to collect additional data or the same data for a longer time span and you can find that the server begins to operate under stressed conditions almost immediately if the hardware is marginal, exposing the data to corruption or degradation. Use the best hardware that you can get management to buy; it will be a bargain over the long run. As an example recovering a large database or several databases hosted on one server can be far more expensive than the cost of the right hardware, that is without counting the loss of business and potential or actual public embarrassment to the Company.
- A fully functional and optimal environment has the following characteristics: documented, automated, standardized, reliable, available and optimized. It is important to document your installation, security and maintenance processes; these lead to, the setup of standardized server installations where every server is configured the same including file locations and backups; which help, in automating the many tasks and delivering a reliable and available system. Since, SQL Server is not a static environment in order to maintain the goals of reliable and available then you need to regularly optimize the server.

- Disaster Recovery and Backups are critical in protecting data, depending on the amount of modification that the data is subjected to one may need to have differential and log backups several times during the business day. SQL Server 2000 has an excellent integrated backup facility that allows for multiple backup options to disk, network disk or tape. Please keep in mind that if your data is highly sensitive, then you should use the backup password feature to prevent unauthorized users from restoring a backup. Keep in mind this feature does not encrypt the data onto the disk device or tape, so physical security of the media is important.
- Secure the underlying Operating System NT or Windows 2000 server as part of the process of installing SQL Server. It is preferred that the securing of the O/S is done prior to the installation due to the possible breaking of the SQL Server setup if done after.

Development

The development of applications that take into account the security of the data is a much more difficult problem now days when few organizations have their own development staffs and the practice is to purchase most applications. Even so, the security and data administration functions have a role in reviewing the purchase candidates for adequate data security functions built into the package, in other cases where the product is in its early stages of development, it is possible to influence the security of the package. I will not try to outline in this paper the methods to build secure SQL applications, instead I will list questions that will aid in determining whether a commercial application is secure or not.

- Is the application accessed via a programmed front end or web page? In cases of applications that use a client program that is written in VB or C usually there is an application security feature that connects to the database, with additional security that is by userid within the database. This is a common method used by most large application packages including SAP. On the other hand if the application is accessed via web pages, then the manner and method of access depends on the setup of a secure web server, along with either Windows Authentication or DLLs that handle the passing of userid to the database server for authentication. It is in this later method that the least secure databases are found in many instances lending themselves to database injection attacks or access by using common desktop applications such as Access and Excel.

- Does the application use a 2 or 3 tier method? It is preferable to implement 3 tiers versus 2 tier applications. The reason why 3 tier works in a more secure fashion is that in this model the front end client GUI communicates with COM+ applications, this isolates users from the data altogether and denies direct communication to the SQL Server, all rules regarding what users can view or modify are managed in the middle tier which can be modified. It also allows for user and client impersonation and authentication with Active Directory or a database stored procedure against a user table within the database.
- If the application uses a web interface is the installation of the web site designed to limit access to selected users? In cases where the application is a web application and the vendor just installs a few ActiveX pages, then the web server should be secured along the recommendations made by SANS in their course *Securing Windows – Track 5 Session 5 Securing IIS* by Jason Fossen. This document outlines how to prepare an IIS server for Internet and intranet use and the steps necessary to secure the same. There are other sources also available on how to secure IIS from Microsoft and others, including a security wizard from Microsoft.
- Does the application require the use of the sa account for installation or operation? In some cases the use of the sa account for installation is required because the vendor has to create the database in that case an account with sysadmins rights on the server can be substituted for sa. In the case where the vendor requires the sa account to operate the application, then a conversation with the vendor is necessary on how to circumvent this problem or the application needs to run from a dedicated server.

Data Security

Securing SQL Server is one of the primary responsibilities of the Database Administrator (DBA). This is where the rubber meets the road, and what separates the professionals from the rest. In many cases the business has to have an application no matter how well it is secured or how poorly, often the application is barely in its Alpha development state or its first Beta version. In some cases when asked about security features the answer from the vendor is “well get to deliver better security in version 3.0” or similar, but it needs to be working in production in less than 60 days. It is the specific training of the DBA that enables him or her to combine all the possible alternatives available to make the application as secure as possible while operational.

- Limit application access to those that have a need to have access to the data. By the use of NT Groups you limit access to the server and to the data even if the application does not make use of Windows Authentication; if the application is accessed via web browser then you can use Windows to authenticate users to the web page, or if using IE5.0 or better or Mozilla 1.0 you can use Digest Authentication to verify that the user belongs to the authorized group of users.
- Limit access to extended stored procedures only to DBA or Domain Administrators use only. There is a list of extended stored procedures that should be restricted from use by regular SQL server users. If an attacker compromises the sql user login (Domain account) using the extended store procedures they can start a command window, add and delete registry entries, change or add SQL user entries, etc. These stored procedures need to be limited so that they can be run only by users that are members of the sysadmins role. A project to help lock down SQL Server has been initiated by Chip Andrews of sqlsecurity.com the goal is to have a series of scripts for SQL Server 2000 which will limit access to the extended stored procedures and apply permissions to account groups more about this can be found at:

<http://www.sqlsecurity.com/DesktopDefault.aspx?tabindex=4&tabid=12>

- Use IPSEC for inter-server communications whenever possible. This limits access only to those users who have a certificate installed and certificates can be issued either by groups of applications or singularly depending on the need to separate the user community. In the past IPSEC has been received poorly in the Windows community but the Windows 2000 implementation and the use of a local certificate server have simplified the use of this protocol. If your applications are 3 tiers and you limit communications between the business process tier and the SQL Server using IPSEC no one without a server certificate can communicate to the SQL box.
- Disable the guest account on production databases. By default the guest account is added to all user databases created in SQL Server. The guest account allows a user to connect to a database without having specific privileges to it. The guest account cannot be deleted but it can be disabled in the user tab of the database in the following manner shown in Figure 3 below:

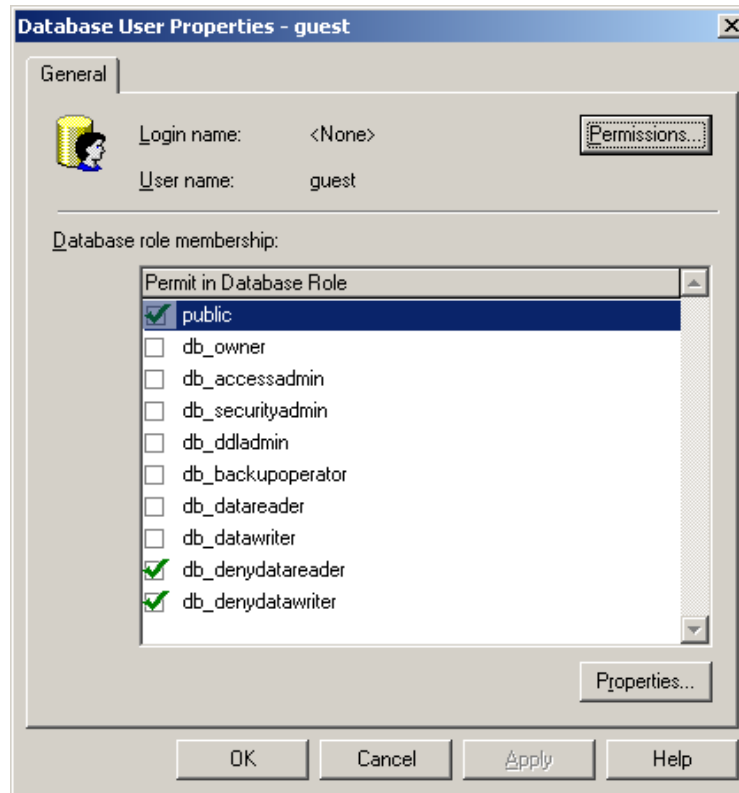


Figure 3

The checkmark on public in Figure 3 cannot be unchecked, but it does not matter since the deny permission takes precedence over the grant to public access.

Auditing

One is not able to determine whether the measures taken in all the previous steps are working unless there is an active audit activity over the SQL Server. Auditing provides the information necessary to establish that the policies and procedures are working as expected. Auditing in SQL Server 2000 comes in two flavors, server auditing and C2 auditing. Server auditing is configured in Enterprise Manager and has four settings: None (default), Success, Failure and All (logs all activity). C2 auditing is enabled from the registry and logs all object activity related to SQL Server, the log is placed on the data directory and a new log is created every 200 MB of log use. If the server runs out of disk space the SQL service will shutdown, until space is made available. SQL Profiler is the tool used to analyze C2 logs, it allows for selection of various types of errors or actions to be filtered from the log file. In the area of auditing the following can help in obtaining the right information:

- If you grant SQL Server access via Windows group membership and you need to trace logins to individual users, then you must use

the SQL Profiler tool to find the userid that made the data modifications. SQL Profiler allows for the selection of the data to be collected.

- In applications that require extensive tracking consider using application auditing. This is normally a part of the table design and can be as simple as the userid of the record creator to the tracking of the userid of every user who has accessed the record.
- If you suspect that someone is trying to access the server in an unauthorized manner use server auditing to track failures and success of logins.
- Keep in mind that the more auditing the greater the impact on the server performance. Balance the need for audit information with the performance impact.
- There are two third party tools worth mentioning if you have a need to do extensive auditing in SQL Server. The first is Omni Audit, this tool sets up triggers to track every change to the user data tables, and records the changes including before and after snapshots of the record in its own separate log files. You can find more information on Omni Audit and a 30 day trial at:

www.krell-software.com

The cost of this product is moderate with a site license under \$1500. The second tool, Lumigent Log Explorer, is a reader-interpreter of the actual SQL Server log files, allowing the user to read the log content with the ability to interpret all the fields of each log entry. It even allows reading logs that were created prior to its installation. The greatest feature of this product is that it allows for the restoration of individual log entries to the database, just like an undo utility. This product can be found at:

http://www.lumigent.com/products/le_sql/le_sql.htm

This product does not have a listed price but is quoted (\$\$\$\$) per named instance.

Conclusion

At the present securing and maintaining the Data Confidentiality, Data Integrity and Data Availability of SQL Server 2000 is a multi discipline set of tasks with a lot of trial and error left to the Sysadmins or DBA responsible. The process of applying patches to SQL Server is far behind the Windows Update available

for operating systems and the Office products today. Security is not applied from one location in a console, but multiple places, some that are well hidden from the untrained administrator. And the ability to analyze the various logs of information provided requires specialized training.

I am told by Microsoft sources that the current product under development, known as Yukon, will have features that will help with many of the hurdles mentioned above, but this new product is still months from its first Beta. There are efforts under way to collect the current knowledge into books such as Young's and to develop scripts and lock down policies such as the effort by Chip Andrews, but we are at an early stage. But as the repository of the most mission critical data in most enterprises, I cannot over emphasize the importance of hardening and securing SQL Server to the highest degree possible and not just waiting for the process to be easier, otherwise you may find yourself restoring databases over a weekend after the next major exploit.

© SANS Institute 2003, Author retains full rights.

Appendix A

Requirements for Domain User Account from Mastering SQL Server 2000 Security Pages 70 –71

If you don't want the account to use local administrator (privileges), you should evaluate the following considerations:

- The account must be able to access and change the SQL Server directory (\Program Files\Microsoft SQL Server\Mssql).
- The account must have access to read and change the .mdf, .ndf, and .ldf database files.
- The account must be assigned the Log On As a Service user right.
- The account must be able to read and write registry keys at and under the following:
 - Hkey_local_machine\software\Microsoft\mssqlserver – also for any additional named instance,
 - Hkey_local_machine\software\Microsoft\Microsoft SQL server
 - Hkey_local_machine\system\currentcontrolset\services\mssqlserver- also for any named instance key, hkey_local_machine\system\currentcontrolset\services\mssql\$instancename
 - Hkey_local_machine\software\Microsoft\windowsnt\currentversion\perflib
- The user account must be able to read and write corresponding registry keys for these services:SQLAgent\$InstanceName (for each named instance on the server), MSSearch, and MDTC.

Additional permissions may be required to gain added functionality in SQL server. Most of this additional functionality enables SQL Server to access and use services outside SQL Server. For example, you may want to configure integration with Exchange Server, so that you may have the option of using the xp_sendmail stored procedure to send mail messages. The following (table) identifies the additional functionality options and the permissions required to perform the additional action.

Appendix A

Additional Functionality	Permission Required
Use xp_sendmail to send mail	Service account needs a mailbox that has permission to send messages
Run xp_cmdshell	User rights of Part of the Operating System and Replace a Process Level Token need to be assigned in Windows 2000
Create cmdexec and active script job steps in a job owned by a non-sysadmin	The Server Agent account must be a local administrator
Add and delete DQL server objects in Active Directory	SQL Server Agent service account must be a member of the local power users group.

Citation of Sources

Andrews, C. ed. "Lockdown Script Project", 26 February 2003 SQLSecurity.com
< <http://www.sqlsecurity.com/DesktopDefault.aspx?tabindex=4&tabid=12>>

Daukantas, P. au. "SQL Server 2000 gets C2 certification; security tests underway for Win 2000" 2 October 2000 Government Computer News, Post Newsweek Tech Media 2000-2003 <http://www.gcn.com/vol19_no29/news/3049-1.html>

Kondreddi, N.V.au. "Overview of SQL Server security model and security best practices" 14 December 2001
<vyaskn.tripod.com/sql_server_security_best_practices.htm>

Litchfield, D. au. "Microsoft SQL Server Passwords (Cracking the password hashes) 24 June 2002 NGSSoftware Insight Security Research 9 pages
<www.nextgenss.com/papers/cracking-sql-passwords.pdf>

Microsoft. Microsoft SQL Server 2000 Operations. Seattle: Microsoft Corporation, 2002.

Nevens, S.C. au. "Database Security – Protecting Sensitive and Critical Information" 31 January 2003, DM Review, 2003
<www.dmreview.com/master.cfm?NavID=198&EdID=6310>

Novick, Andrew au. "15 seconds: Securing SQL Server for Web Applications" Internet.com, 15 July 2002 <www.15seconds.com/Issue/020715.htm>

Peikari, C. and S. Fogie au. "Guarding Against SQL Server Attacks: Hacking, cracking, and protection techniques." Airscanner.com 2002
<www.airscanner.com/publications.html#articles> pdf file.

"SQLSecurity Checklist". SQLSecurity.com Check list for securing SQL Server
<www.sqlsecurity.com/DesktopDefault.aspx?tabindex=3&tabid=4>

Spencer K. au. "Using Windows Security with IIS and SQL Server 2000" 14 November 2001 Windows and .NET Magazine, Penton Media Inc. 2003
<www.winnetmag.com/WindowsServer2003/Index.cfm?ArticleID=23035>

Talmadge, R. au "Securing Your SQL Server" DevX Inc., 2003
<www.devx.com/codemag/article/10290>

Waymire, R. and B. Thomas au. "Microsoft SQL Server 2000 Security" October 2001 Microsoft Corporation 2001 – 2003
< http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsq12k/html/sql_security2000.asp>

Weidman, B. au. "Database Security (Common-sense Principles)"
Governmentsecurity.org<www.governmentsecurity.org/articles/DatabaseSecurityCommon-sensePrinciples.php>

Young, M., and C. W. Young. Mastering SQL Server 2000 Security. Indianapolis, Indiana: Wiley P, 2002.

© SANS Institute 2003, Author retains full rights.