Global Information Assurance Certification Paper

Anthony D Bender
GSEC Version 1.4b
SQL Server 2000 – Protect your data

Abstract

The Sapphire and SPIDA SQL worms have opened the eyes of many administrators world wide. These worms will be reviewed in more detail to open those eyes even wider. Once the vulnerabilities are reviewed, solutions are then given to secure SQL servers. Advice is given on where to place a SQL server in an environment that would support a website that gets its data from a SQL Server. A section on how to get SQL server patched and how to tell what version you are running is available. A solution is provided to encrypt the rows and columns of a database. This will provide an extra layer of protection for data like credit card numbers and social security numbers. Authentication with SQL server is explained and gives tips on when to use SQL server authentication and when to use Windows Authentication. Tools of the trade are defined and examples are given to familiarize the reader with them.

The Threats – recent worms

Recently, worms have surfaced on the internet that are designed to attack SQL Servers, in particular SQL Server 2000. Another recent worm was the SQL SPIDA worm. Both of these worms used SQL Server to perform Denial of Service attacks on random addresses on the internet. The following section will go over these two recent worms in more detail.

Sapphire SQL Worm

As I mentioned earlier SQL administrators and network security personnel were awakened by the Sapphire SQL worm on January 25, 2003. I was up late that night working when I noticed a lot of traffic coming from what looked like all over the internet on our firewall. Here is a small sample from one of our firewalls right when the traffic first started:

Jan 25 00:29:46.339 XXX gwcontrol[0]: 201 1434/udp[72657608]: access denied for xxx.xxx.xxx.xxx to 192.168.xxx.xxx [default rule] [no rules found]

Jan 25 00:29:46.339 XXX udp-gsp[808]: 121 Statistics: duration=0.00 srcif=5B32E034-FA0D-4 src=xxx.xx.xxx.xx/4124 cldst=xxx.xxx.xxx.xxx/1434 svsrc=xxx.xxx.xxx.xxx/4124 dstif=00842333-8964-4 dst=xxx.xxx.xxx.xxx/1434 proto=1434/udp (Not authorized)

This traffic started to come in at around 12:30 am on January 25th, 2003. Our systems remained stable because we do not allow SQL Server ports 1433 and 1434 tcp and udp in from the internet, but many businesses were affected by this worm.

120,000 computers were infected within the first few minutes according to News.Com. By early morning administrators all over the world were cleaning up after the mess. I myself was not affected except for an extra large firewall log for a Saturday. Even though we deny everything unless there is a rule to allow, I wrote a deny rule that denied all traffic for ports 1434 udp and tcp on our outside interface.

On the morning of January 27th 2003 I installed SQL Server Service Pack 3 after some testing on non production servers. This worm could have been much more disastrous than it was. It seems there was no malicious intent other than a denial of service attack. This attack could have easily been done during the day on a Wednesday morning and could have resulted in havoc all over the internet. Better yet it could have been more than a denial of service attack like the SQL Spida worm was. I would support the idea that a "White Hat" hacker was doing his or her ethical part to keep systems and data from being compromised.

An SQL server vulnerability was found, a patch was released to fix this hole and SQL Server 2000 service pack 3 was released January 17, 2003. Service pack 3 also had the fix for the problem. A lot of administrators did not apply the patch or the service pack as we have come to find out today. The Sapphire SQL Worm surfaces just one week after the service pack was released. The following is an excerpt from an article By Robert Lemos Staff Writer, CNET News.com, named "Setbacks in search for worm Author".

"Some experts have focused on Hong Kong. Security research firm iDefense on Monday released an analysis that suggested the worm may have originated from the Chinese hacking group Honker Union of China (HUC), but the company admitted that the evidence is slim and mostly circumstantial.

"A definite correlation between the actual worm code and the HUC is still undetermined," iDefense analysts wrote in the report. "As far as is known, the group has not claimed responsibility for the attack."

The company based its analysis on the fact that the hacking group--which was responsible for many Web site defacements following the U.S.-China spy plane incident in April 2001--had posted code capable of exploiting the flaw in Microsoft's SQL Server. iDefense pointed to the fact that the code began with a "no operation," or NOP, command and that the group had a member whose handle was n0p."

Assume that the group from China is not responsible. A group of white hat hackers determine that there is a great risk associated with the vulnerability and decide to help fix the hole by writing a worm that would be noticeable enough to slow the internet down but not bad enough to cause major outages and data loss. This gets the vulnerability headline news. Servers and holes get fixed without massive outages and data loss that could have been caused by HUC or some other malicious hacker. This theory supports ethical hacking. I do not think

that the writer of this worm should be accused of causing denial of service on some ISP's and business networks late on a Friday night or Saturday morning depending on where you are in the world. He should be praised for his efforts to secure a major vulnerability. If this had been done in greater force by Iraq at the same time that strategic terrorists attacks on major cities occur it would cause major economic failure and would send the American people into frenzy. Thanks to the writer of Sapphire for doing your part. A different spin on this theory would be the possibility that an attacker wanted to draw attention to this vulnerability to get the world's heads turned in the other direction while he would go after a single target worth millions.

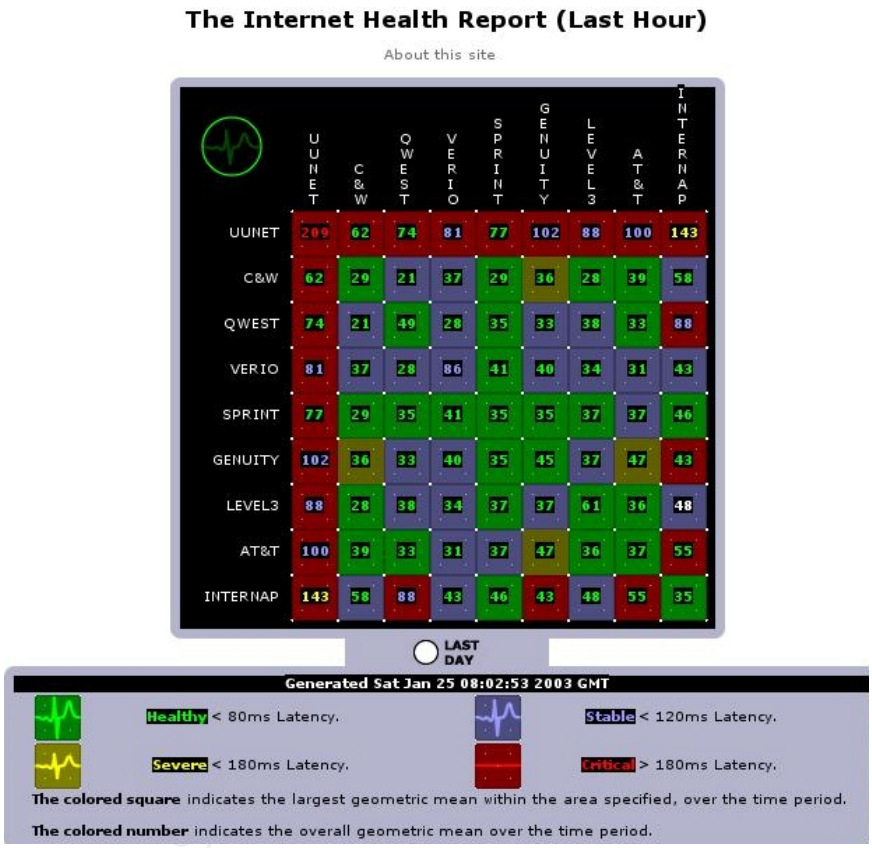

Figure 1

The worm uses a buffer overflow to exploit a flaw in Microsoft SQL Server 2000. The SQL Server 2000 flaw was discovered in July, 2002 by Next Generation Security Software Ltd. The buffer overflow exists because of the way SQL Server improperly handles data sent to its Microsoft SQL Monitor port. Attackers leveraging this vulnerability will be executing their code as SYSTEM, since Microsoft SQL Server 2000 runs with SYSTEM privileges, according to Marc Maiffret of eeye.com. A full analysis of the worm can be found at

http://www.techie.hopto.org/sqlworm.html by Matthew Murphy. Its affect on the internet is shown in figure 1 and figure 2 taken from http://www.silicondefense.com/research/sapphire and www.cyberarmy.co.kr/~corea2k/worm . Notice that at 5:00am on Saturday there were no infections and then at 6:00 am most of the advanced portions of the world were infected. You will get an idea now on just how quickly this worm spread.

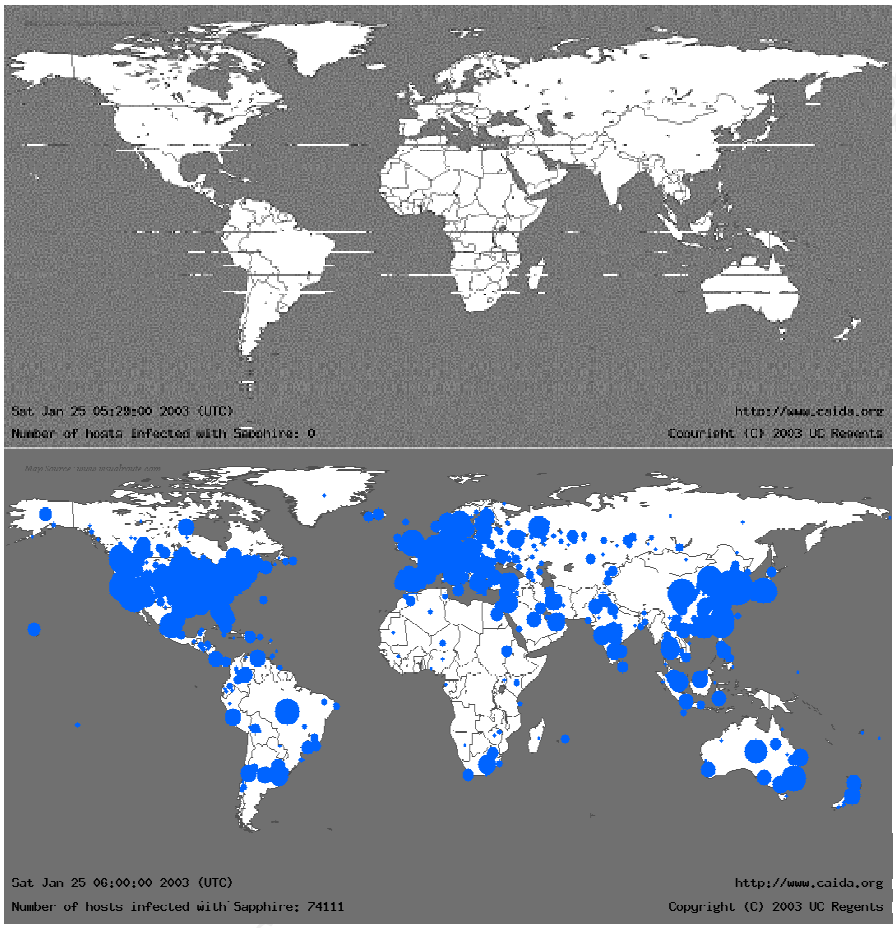

Figure 2

SQL-Spida-Worm

The SQL-Spida worm first hit the internet on May 21st 2002. It has not infected as many computers as the Sapphire worm did, only about 2,865 since it first appeared according to wtc.trendmicro.com. Figure 3 shows how this worm started out very powerful but died down quickly.

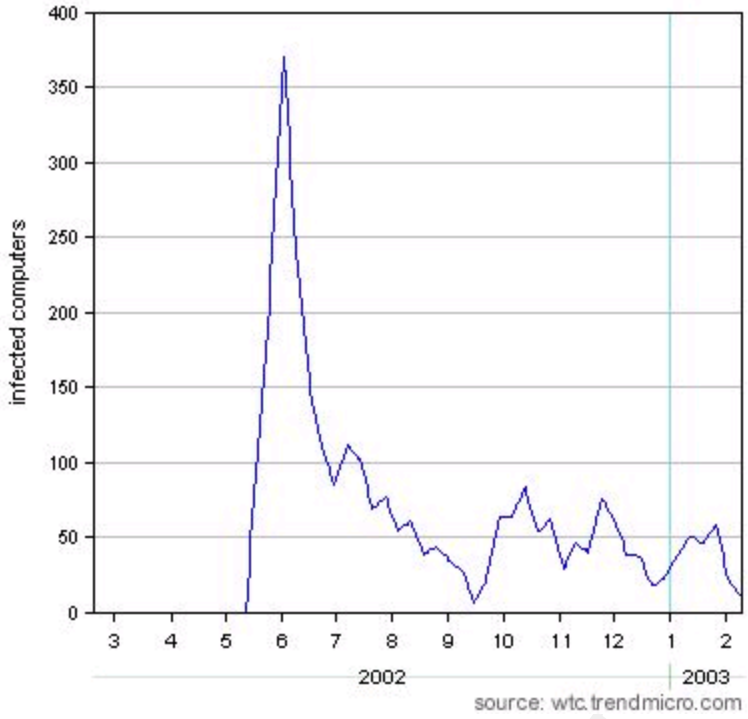

Figure 3

This worm takes advantage of SQL servers that have no system administrator or sa password. It actually opens a command shell on the computer that it has found, by running the xp_cmdshell. This extended stored procedure is used to execute system commands within SQL queries. Once the command shell is running the following files are copied to the windows/system32 directory.

- %SysDir%\DRIVERS\SERVICES.EXE
- %SysDir%\SQLEXEC.JS
- %SysDir%\CLEMAIL.EXE
- %SysDir%\SQLPROCESS.JS
- %SysDir%\SQLINSTALL.BAT
- %SysDir%\SQLDIR.JS
- %SysDir%\RUN.JS
- %SysDir%\TIMER.DLL
- %SysDir%\SAMDUMP.DLL
- %SysDir%\PWDUMP2.EXE

The sqlprocess.js file is the main file for this worm. It contains the ip address arrays which are later used by the services.exe to scan for other machines. With the command shell open and ready to use it does an ipconfig/all and puts the address information into a send.txt file. Pwdump2 is then executed and when it is done getting the hashes on the infected machine it puts those in the send.txt file also. You may wonder how the worm got access to the SAM database on the machine. Well when the “sa” password is left blank we already

know that a command shell is available and any commands put in are run as system. The “sa” account in SQL Server runs as system on the computer on which SQL Server is installed. Once the IP address information and the hashes are loaded into the send.txt file it is emailed to ixltd.postone.com. Once the email is sent, the send.txt file is deleted. This worm then uses services.exe to find more SQL servers that are vulnerable to this attack of a blank sa password. The email address that the worm sent information to, had its quota exceeded in the first day. Why the worm sent its information it collected to a single address is a little weird. One would think that the writer would have opened an FTP connection to 1 of the 1000 machines that he owned and sent the information there. He would have collected much more data that way. Maybe the writer knew full well what he was doing and did this on purpose. Surely once the email account exceeded the quota, there were many hackers sniffing the internet for these emails trying to collect the data for themselves.

This worm demonstrates how an unprotected SQL server that has a blank “sa” password can be very beneficial to an attacker both for gaining information about the server infected and also for attacking other computers on the internet. Securing the SQL servers is a must. Once again I must think that these SQL servers were being used by attackers all along. A “White Hat” hacker that supports ethical hacking as I do, saw this activity and wrote a worm that would make world headline news. Once the attention was gained SQL server “sa” passwords were fixed and data was protected. I may be wrong on this, but I can not believe that these SQL servers were discovered by a single, well written worm. They had to have had been previously discovered by hackers. I believe that out of the 2,865 computers that were infected some of those had already been discovered. A detailed overview of this worm can be found at the following URLS. http://www.iss.net/security_center/alerts/advise118.php

http://www.nipc.gov/warnings/advisories/2002/02-003.htm

“Defense in Depth”

In the computing world today there are web sites that are interactive with the user. For example, you start up your browser and log into abcbank.com and use the online banking system. You may also use the banks bill pay system to manage your bills. You have all of your payee information including account numbers. Think of all the personal information that is stored in their possibly unprotected databases. All of your credit card information, mortgage loan information, car loan information and all the companies to which you pay bills. This is a very serious issue because abcbank.com has to protect this information. This information is more than likely stored in a SQL Server. In a report by Robert Lemos entitled “Spida spreads venom on MS SQL Servers” he writes “SQL Server is the third most widely used database software package, with more than 16 percent of the overall market, according to a recent survey by Gartner Dataquest.” There is a real need for network security personnel to secure those SQL Servers.

At the following URL http://www.microsoft.com/mscorp/execmail/default.asp you will find Bill Gates' executive email entitled "Security in a Connected World" released January 23, 2003. Bill Gates states the following "As everyone who uses a computer knows the confidentiality, integrity and availability of data and systems can be compromised in many ways, from hacker attacks to Internet-based worms. These security breaches carry significant costs. Although many companies do not detect or report attacks, the most recent computer crime and security survey performed by the Computer Security Institute and the Federal Bureau of Investigation totaled more than $455 million in quantified financial losses in the United States alone in 2001. Of those surveyed, 74 percent cited their Internet connection as a key point of attack." The threat is real.

We need to protect our data better. Do not be so foolish as to think that setting complex passwords and having a password policy is enough. If SQL Servers are left out in the open they will be hacked. It has become my belief that critical databases should be behind a separate firewall as shown in figure 4. If you are hosting a web site and the web server gets its data from a SQL Server it should be behind another firewall besides your perimeter firewall. No traffic should be redirected in from the perimeter firewall to any port on your SQL Servers. There should be no reason that a website user on the internet needs to communicate directly to your SQL server. Redirect port 80 or 443 into your web servers and then let the web servers communicate with the SQL Server. The web server and SQL server should be different machines. Never host a website on a SQL Server unless you do not care about the data that is on it. Also it is a good practice to put an IDS, Intrusion Detection System, on the three segments as shown in the figure 4. Let's say for some reason the web server in figure 4 gets compromised. If your SQL Server was on that same segment, it would be much easier for an attacker to gain access. Since it is behind another layer of protection it would be much more difficult to gain control over. It is also less likely that all three IDS nodes and the 2 firewalls would miss a break-in, but if they did, the attacker would be able to sniff the traffic from the switch. Figure 4 shows that the web servers and the SQL servers are plugged into the same switch. This could pose a problem; a solution is to encrypt the traffic between the web server and the SQL server. A way to solve the problem of an attacker being able to sniff the network for packets going between the SQL Server and the web server, would be to dual NIC the web servers and separate the traffic all together. When using the dual NIC scenario the firewall could be locked down to the Mac addresses of the web servers. Rules could be written to allow only SQL traffic between them.

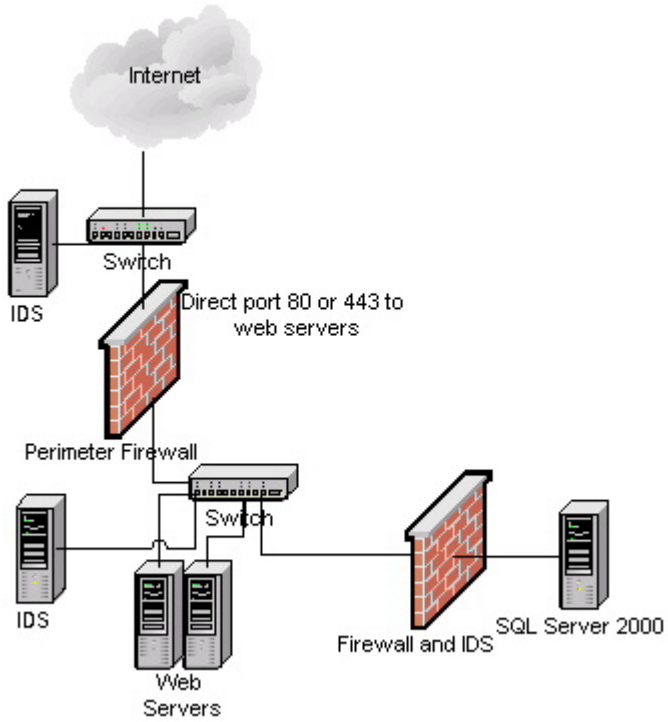

Figure 4

Get Patched

In order to prevent known vulnerabilities always keep up with the latest patches and service packs; SQL Server is currently up to service pack 3. Recently, due to the Sapphire SQL worm, myself and many other administrators have learned that not keeping up with patches can be very disastrous. A free script from http://www.sqlsecurity.com tools section named version.sql can be run from the query analyzer to let you know exactly what version of SQL Server you are running. This will assist you in figuring out which patches you need to apply. Another way to tell what version you are on is by running the following from the query analyzer: "select @@version" The following is what is returned on a SQL server that has service pack 3 installed. Microsoft SQL Server 2000 - 8.00.760 (Intel X86) Dec 17 2002 14:22:05 Copyright (c) 1988-2003 Microsoft Corporation Enterprise Edition on Windows NT 5.0 (Build 2195: Service Pack 3) If your SQL Server is not patched you can compare the results to the list found at http://www.sqlsecurity.com checklist section. This is also a very good checklist to assist you in locking down SQL Server. It will go over what stored procedures are a security risk and can be deleted along with many other lock down tips.

Data Encryption

Another layer of protection is encryption. There is a site http://www.activecrypt.com which has a product that is fairly inexpensive that will encrypt data at the column and row level. Its new version uses an MD5 hash. According to activecrypt.com, "customers who are using the product are financial, medical, internet service providers, industrial companies, banks, solution providers and entertainment sites. They have customers all over the USA, Canada, Europe, Australia and Asia. Their

customers are different in budget: from small hosting providers up to companies listed in NASDAQ." Here is a list, taken from their site, of encryption levels that it will handle.

	Free version	Full version
DES hash	**Unlimited**	**Unlimited**
SHA1 hash	**Unlimited**	**Unlimited**
RSA key generation	**Fixed to 256 bits**	**Unlimited**
RSA encryption	**Limited to 21 chars**	**Unlimited**
AES	**Limited to 15 chars**	**Unlimited**
DESX	**Limited to 15 chars**	**Unlimited**
RC4	**Limited to 15 chars**	**Unlimited**
Loading x.509 certificates	**No**	**Yes**

Figure 5

Another program that can encrypt data at the column and row level is DBEncrypt and can be found at http://www.appsecinc.com/products/dbencrypt/mssql/ .

A defense in depth approach to protecting our data is a must and it is not hard to protect just takes diligence. Computer crime in 2001 cost the United States $455 million in quantified financial losses as Bill Gates stated earlier. This is not acceptable.

Authentication

In SQL Server 2000 there are two ways that a user can be authenticated to use the database. The first and less secure way is SQL Server Authentication. "SQL Server uses an undocumented function, pwdencrypt() to produce a hash of the user's password, which is stored in the sysxlogins table of the master database. This is probably a fairly common known fact. What has not been published yet are the details of the pwdencrypt() function." Litchfield

To get the hash of the sa password on a SQL Server do the following. I will demonstrate this using an installation of SQL that is not in a production environment. Bring up Query Analyzer and run the following:
select password from master.dbo.sysxlogins where name='sa' or any other username in the server that you set up. I have made my password "pack3t!" Here is the hash that is returned from the query:

0x01002F2E37344803F25011DC862F36B97C5F630F14C8963696C8A9B5D28
3E984CCEA3CF3255D7CBD8D266F3B88AB

To understand SQL Server password hashes and how it stores, creates and uses passwords I suggest you read the following report "Microsoft SQL Server

Passwords Cracking the Password Hashes" by David Litchfield at the following URL http://www.nextgenss.com/papers/cracking-sql-passwords.pdf

The second way that SQL Server can authenticate a user is through Windows Authentication. There have been discussions on which authentication method is more secure. Windows Authentication needs a domain to which to send authentication requests. With no domain, Windows Authentication would not be feasible. Another situation to be aware in regards to authentication is that just because there is a password policy in place does not keep the data safe. A company has a password policy that states that domain passwords must be 8 or more characters in length, have to meet complexity requirements and passwords expire every 45 days. It states that under no circumstances should your password be given to anyone and if it becomes compromised it must be changed. If this is the case then using Windows Authentication to let user's access databases would be more secure than using SQL Server Authentication. On the other hand if there is no Win2k active directory domain to authenticate users passwords or if there is a weak password policy in place then SQL Server Authentication could be used.

Some do and don'ts when using SQL Server Authentication are:

1. Always set an sa password. When installing SQL Server it gives you the option to leave it blank, although it says that it is not recommended. Administrators have, for some reason or another, left the sa password blank, the SPIDA SQL Worm proved that. This gave the worm and hackers a command shell on the box as we discussed earlier.
2. When using SQL Server Authentication in web page code never use the sa password always create a new user or role and give that user or role the least amount of access that is needed. Always give the user a password. There is also a solution to making sure that users passwords have some complexity. This will be discussed later.
3. Audit your SQL passwords regularly, just as much as you would audit your Active Directory passwords. Tools to audit SQL passwords are available This will be discussed later.
4. Enforce complexity in your SQL passwords. There are also tools that will do this for you, these are mentioned in the tools section of this report.

Tools

This section is going to go over some of the tools that make securing SQL Server easier to accomplish. I will break them up according to category. The first category will be on passwords. NGSSQLCRACK can be found at http://www.nextgenss.com and is a tool that is used to audit passwords. It is fairly fast and can only be run by the sa and local administrator users since their accounts are the only accounts with access to the password hashes. When you bring the program up you will need to get the hashes from the server you are auditing. To get the hashes you will have to supply the server address or name

and a username and password that has admin access. Once the hashes are retrieved, passwords can be audited. A plain text password of “sqlserver” was cracked in 5 seconds once the hash was loaded into the program on my test system. A password !@#$%^&*()0987654321 was not cracked at all. Force SQL v2.0 can be found at http://www.nii.co.in and can also be used to accomplish the same thing.

So are you paranoid about your user passwords now? Then this next tool is for you. “Enforcepass” enforces complexity to SQL passwords and is found at http://www.nii.co.in under the research/free tools section. Instructions come with the download file and implementing this is only a matter of running a few SQL scripts from the Query Analyzer. Once installed, the software ensures the following when a user creates a password.

1. It must not be the same as the login name.
2. It must not contain the login name.
3. Its length must be greater than 6 characters.
4. It must contain at least one alpha-character.
5. It must contain at least one numeric-character.
6. It must contain at least one punctuation character

For encryption of databases, stored procedures, rows and columns use the program that was mentioned earlier found at www.activecrypt.com called xp_crypt. This program is easy to use. You must make sure the encryption keys are backed up in case recovery is needed. If the data is encrypted and something happens to the keys then there is no way to get the data back to normal. Disregarding your keys with this program can be disastrous.

To make sure there are no runaway SQL servers in your network, there are a multitude of ways to find them. You can use SQLPingv2.2 and SQLPoke. These two are scanners that scan the address range that is entered in for SQL servers that are answering on ports 1433 and 1434, which are the well known ports that SQL Server uses. Also from the site www.eeye.com vulnerability checking programs can be downloaded. For example, when the Sapphire SQLworm came out that Monday, eeye.com had come out with a GUI that would check for the vulnerability. It would scan the address range entered and then display whether or not the server was patched or not. Free scanning tools that scan for the SPIDA worm can also be downloaded from their site. You can also use Nessus to scan a range of addresses and only load the SQL server plug-in. Nessus can be found at www.nessus.org Nmap can be used to scan for addresses that answer on tcp/udp 1433 and 1434. One could use a sniffer like ethereal and NGSSniff from www.nextgenss.com to sniff the network for traffic that is using or answering on ports tcp/udp 1433 and 1434.

Once you have found these servers you can use NGSsquirrel from www.nextgenss.com to help lock them down. This software can be run against a

SQL server and it will determine security holes. The following is what NGSSquirrel came up with on my test server which has nothing done to it except service pack 2. The evaluation version only tells you of the problems that it found. The full version will generate a SQL Server lockdown script to run that will lock it down. I recommend doing this a couple times on some test boxes before jumping into live servers since it may cause more harm then good. Be familiar with your applications and your SQL Server before applying the recommendations that is gives. All and all this program is very beneficial to have in order to be able to see what state your SQL servers are in and get them fixed and protected. You will probably be surprised how much it will find and the good thing is that it explains what the problem is, similar to how Nessus will. This software will pay for itself. NGSSQuirreL saves the SQL Server administrators a lot of time. It would take them twice as long to do the lockdown procedure manually using the lockdown checklist mentioned earlier.

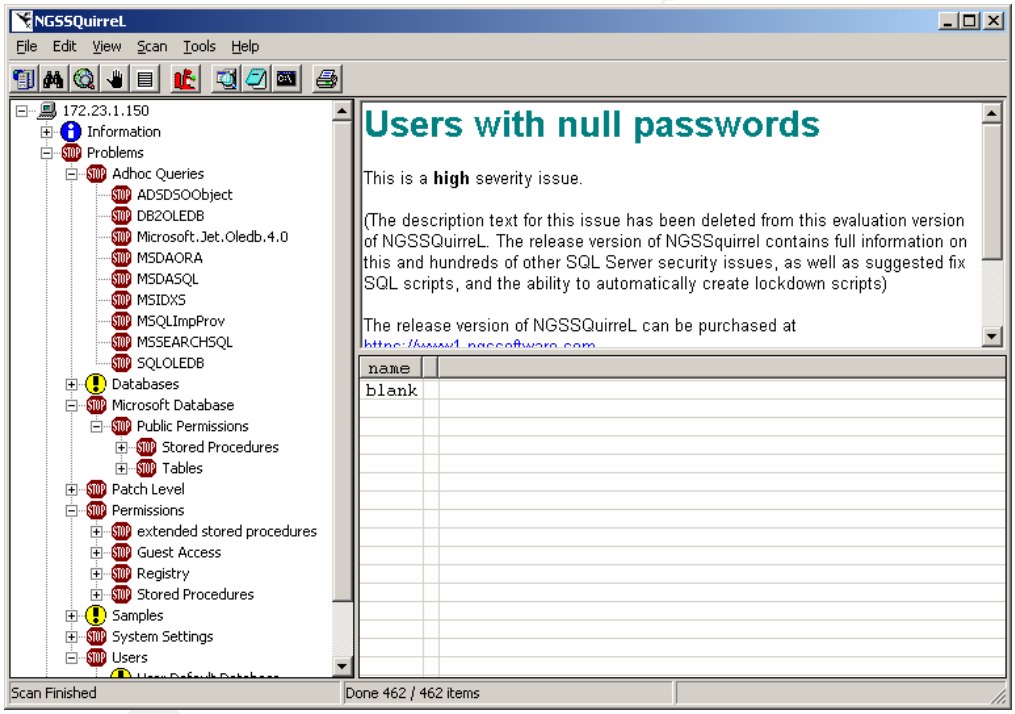

Figure 6

Conclusion

In conclusion, I have identified the threats and given advice on how to secure your SQL Servers. I have mentioned tools that are helpful in this process as well. I would like to mention that an enterprise can have all the sophistication and technology in the world but bottom line is that the data is secured by the administrators and network security engineers. These administrators and engineers need to be diligent in their work. They need to try new products, read magazines and then practice what they have learned. I challenge the SQL

administrators of our businesses today to take note of what was written about in the paper, take it to heart and secure your SQL Servers today.

References

Lemos, Robert Staff Writer Cnet News.com "Setbacks in search for worm Author" January 27, 2003 http://news.com.com/2100-1001-982284.html

Lemos, Robert ZDNet News "Spida Spreads venom on MS SQL Servers" May 21, 2002 http://zdnet.com.com/2100-1105-920187.html

Murphy, Matthew "Analysis of Sapphire Sql Worm" http://www.techie.hopto.org/sqlworm.html

David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, Nicholas Weaver "The Spread of the Sapphire/Slammer Worm" http://www.silicondefense.com/research/sapphire/

Internet Health Picture during Sapphire Worm http://www.cyberarmy.co.kr/~corea2k/worm/

"SQL-SPIDA-WORM (9124)" May 21,2002 http://www.iss.net/security_center/static/9124.php

"Microsoft SQL Spida Worm Propagation", May 21,2002 http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise118

Microsoft "Finding and Fixing Slammer Vulnerabilities" February 7,2003 http://www.microsoft.com/security/slammer.asp

Peters, Peter "Microsoft SQL Spida Worm Propagation" 22-May-2002 http://cert-nl.surfnet.nl/i/2002/I-02-01.htm

Bakos, George "SQLsnake Code Analysis" 21 may 2002 http://www.incidents.org/diary/diary.php?id=157

" *"Spida" worm on Microsoft SQL-Server systems"* http://www.cert-ist.com/english/advisories/avis_spida_en.htm

Trend M icro Virus Overview SQL-SPIDA http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=JS_SQLSPIDA.B&VSect=S

Gates, Bill. “Security In a Connected World” January 23, 2003
http://www.microsoft.com/mscorp/execmail/default.asp

Litchfield, David “Microsoft SQL Server Passwords Cracking the Password Hashes http://www.nextgenss.com/papers/cracking-sql-passwords.pdf

Application Security Inc. DbEncrypt Software
http://www.appsecinc.com/products/dbencrypt/mssql/

Active Crypt XP_Crypt 3.4 Encryption
http://www.activecrypt.com

SQL Security Checklist, and Free Tools
http://www.sqlsecurity.com

NGSSoftware “NGSSQLCrack” and “NGSSquirrel”
http://www.nextgenss.com

Retina Sapphire and SPIDA worm scanners
http://www.eeye.com

Lewis, Morris “Guard Your Data With Kerberos”
SQL Server Magazine July 2002

McClure, Stuart , Shah, Saumil , Shah, Shreeraj “Web Hacking Attacks and Defense” Chapter 2 and 11 Published July 2002