

[NT] Microsoft SQL Server 2000 Unauthenticated System Compromise

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-07/0115.html>

From: support@securiteam.com

Date: 07/25/02

From: support@securiteam.com

To: list@securiteam.com

Date: Thu, 25 Jul 2002 13:33:09 +0200 (CEST)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

Microsoft SQL Server 2000 Unauthenticated System Compromise

SUMMARY

Microsoft's database server SQL Server 2000 exhibits two buffer-overrun vulnerabilities that can be exploited by a remote attacker without ever having to authenticate to the server. What further exacerbates these issues is that the attack is channeled over UDP. Whether the SQL Server process runs in the security context of a domain user or the local SYSTEM account, successful exploitation of these security holes will mean a total compromise of the target server and its data.

DETAILS

SQL Server can be configured to listen for incoming client connections in several different ways. It can be configured such that clients can use named pipes over a NetBIOS session (TCP port 139/445) or sockets with clients connecting to TCP port 1433 or both. Whichever method is used the SQL Server will always listen on UDP port 1434. This port is designated as the Microsoft SQL Monitor port and clients will send a message to this port to dynamically discover how the client should connect to the Server. This message is a single byte packet, the byte being 0x02.

Securiteam: [NT] Microsoft SQL Server 2000 Unauthenticated System Compromise

There are other messages that can be sent to this port and these can be worked out with simple experimentation.

Stack Based Buffer Overflow:

When SQL Server receives a packet on UDP port 1434 with the first byte set to 0x04, the SQL Monitor thread takes the remaining data in the packet and attempts to open a registry key using this user supplied information. For example, by sending \x04\x41\x41\x41\x41 (0x04 followed by 4 upper case 'A's) SQL Server attempts to open

```
HKLM\Software\Microsoft\Microsoft SQL
Server\AAAA\MSSQLServer\CurrentVersion
```

By appending a large number of bytes to the end of this packet, whilst preparing the string for the registry key to open, a stack-based buffer is overflowed and the saved return address is overwritten. This allows an attacker to gain complete control of the SQL Server process and its path of execution. By overwriting the saved return address on the stack with an address that contains a "jmp esp" or "call esp" instruction, when the vulnerable procedure returns the processor will start executing code of the attacker's choice. At no stage does the attacker need to authenticate.

Heap Based Buffer Overflow:

When SQL Server receives a packet on UDP port 1434 with the first byte set to 0x08 followed by an overly long string, followed by a colon character (:) and number a heap based buffer is overflowed. As this corrupts the structures used to keep track of the heap, an attacker can overwrite any location in memory with 4 bytes of their own choosing. This can be used to gain remote control of the processes execution. If the colon and number are missing, the SQL Server process access violates before the heap is corrupted as the code in the SQL Monitor thread fails to handle exceptions.

For example the code calls the C function strtok(). The strtok() functions looks for a given token in a string, in this case a colon, and if found returns a pointer to it. If the colon is missing in the string being searched then no pointer is returned. This is one of the reasons why the SQL Server process access violates if the colon is missing. The code does not check to see if a valid pointer has been returned before passing it to another function call, atoi():

```
char *ptr=NULL;
int num=0;
.
ptr = strtok(string,":");
num = atoi(ptr); // ptr is used without being validated
```

Failure to check return values and handle exceptions leads to the process dying, leading to a simple Denial of Service attack. That said, in the light of the overflows, the DoS is the least of the problems.

Securiteam: [NT] Microsoft SQL Server 2000 Unauthenticated System Compromise

Network Based Denial of Service:

When an SQL Server receives a single byte packet, 0x0A, on UDP port 1434 it will reply to the sender with 0x0A. A problem arises as SQL Server will respond, sending a 'ping' response to the source IP address and source port. This 'ping' is a single byte UDP packet – 0x0A. By spoofing a packet from one SQL Server, setting the UDP port to 1434, and sending it a second SQL Server, the second will respond to the first's UDP port 1434. The first will then reply to the second's UDP port 1434 and so on. This causes a storm of single byte pings between the two servers. Only when one of the servers is disconnected from the network or its SQL service is stopped will the storm stop. This is a simple network based DoS, reminiscent of the echo and chargen DoSes discussed back in 1996 (<http://www.cert.org/advisories/CA-1996-01.html>) (<http://www.cert.org/advisories/CA-1996-01.html>). When in this state, the load on each SQL Server is raised to c. 40 – 60 % CPU time.

Considerations for protection against these vulnerabilities:

Exploitation of these security holes goes over UDP, a connection-less communications protocol. As such, it makes the task of bypassing the protection offered by a firewall considerably easier. The spoofing of an IP address in a UDP packet is also considerably easier.

It is trivial for an attacker to send an attack through the firewall, setting the source IP address to that of the target's DNS Server and the source port to 53. Most firewalls will allow this packet through, as it will look like a response to a query to resolve a domain name.

It is strongly recommended that a rule be added to each organization's firewall such that any packet destined for UDP port 1434 on the 'clean' side of the firewall be dropped and logged. No host, even DNS Servers, should be allowed to send traffic to this port.

It is also recommend that firewall administrators ensure that any packet received on the 'dirty' interface with a source IP address set to an address on the clean side is also dropped and logged.

Fix Information:

NGSSoftware alerted Microsoft to this problem on 17 May 2002 and they have produced a patch that resolves these issues. NGSSoftware urge all customers of SQL Server 2000 to test then apply these fixes as soon as possible.

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp>
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS02-039.asp>

Where possible, NGSSoftware also recommend running the SQL Server as low privileged local account and not SYSTEM or a domain account.

ADDITIONAL INFORMATION

Securiteam: [NT] Microsoft SQL Server 2000 Unauthenticated System Compromise

The information has been provided by <mailto:nisr@nextgenss.com>
NGSSoftware Insight Security Research.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

-
- **Previous message:** support@securiteam.com: "[\[NT\] Authentication Flaw in Microsoft Metadirectory Services Could Allow Privilege Elevation](#)"
 - **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)