

Unauthenticated Remotes in Database Server Software

Making comparisons about database security between one vendor's offerings against another can be a tricky game. On one hand we're told in marketing material that this or that database is unbreakable, or another has passed umpteen independent security evaluations – but on the other hand there are cold facts: database server software has bugs like all software and some of these holes are big enough to drive a bus through. What is a good measure then by which to compare different database servers? This paper will examine the most critical of security flaws – those that allow a remote attacker to gain access to a database server without a user ID or password – otherwise known as the Unauthenticated Remote. The premise is simple – we might consider the least secure database server software as the one with the most holes of this kind and the most secure as the one with the least holes. But is this actually accurate? Just because database X has no such holes – does it mean it's secure? Or does it mean that no-one's really bothered to check, yet? Or does it mean that one database server has more attack surface and another less? This is nigh on impossible to answer and any conclusions drawn by the reader must be their own. What we present here is that which is known – the list of Unauthenticated Remotes that have so far been made public.

MySQL Authentication Bypass 1
Discoverer: Robert van der Meulen, 8 th February 2000
Advisory: http://seclists.org/lists/bugtraq/2000/Feb/0134.html
Affects: MySQL 3.22
When connecting to MySQL, if a client sent only 1 byte of the password hash only one byte would be checked – thus an attacker need try only 32 possibilities to gain access.
MySQL Authentication Bypass 2
Discoverer: Chris Anley, 1 st July 2004
Advisory: http://www.nextgenss.com/advisories/mysql-authbypass.txt
Affects: MySQL 4.1 and 5
When connecting to MySQL, if a client issued specific capabilities they could control the number of bytes of the password hash to be checked by the server. This is almost exactly the same as the “MySQL Authentication Bypass 1” but re-introduced.
MySQL Long Password Buffer Overflow

Discoverer: Chris Anley, 1 st July 2004
Advisory: http://www.nextgenss.com/advisories/mysql-authbypass.txt
Affects: MySQL 4.1 and 5
By passing a long scramble string a stack base buffer can be overflowed. This overflow is difficult to exploit in practice as the buffer is filled with characters in the range of 0x40 to 0x5F.

IBM's DB2

DB2 JDBC Applet Server Username and Version Buffer Overflow
Discoverer: David Litchfield
Advisory: http://www.nextgenss.com/advisories/db205012005D.txt
Affects: DB2 Universal Database 8.1
The DB2 JDBC Applet Server allows java clients to query the database server and it suffered from a buffer overflow vulnerability. Firstly, the attacker would send a long username then disconnect. Next, the attacker would reconnect and send a long version identifier – this would trigger the overflow during the logging of an error.

IBM's Informix

Informix Long Username Buffer Overflow
Discoverer: David Litchfield
Advisory:
Affects: Informix Dynamic Server 9x
Informix suffers from a buffer overflow vulnerability when passing an overly long username. This affected all operating systems and the beta version of IDS 10.

Microsoft SQL Server

Microsoft SQL Server 2000 Hello Bug
Discoverer: Dave Aitel, July 2002
Advisory: http://www.microsoft.com/technet/security/bulletin/MS02-056.msp
Affects: Microsoft SQL Server 2000
When a client connects to a Microsoft SQL Server system the first thing it does is pass a version string. If this string is too long then a buffer overflow vulnerability can be exploited. This bug was named the “hello” bug in reference to the line from the film “Jerry Maguire”, “You had me at hello.”

Microsoft SQL Server Resolution Buffer Overflow
--

Discoverer: Dave Litchfield, 25 th July 2002
Advisory: http://www.nextgenss.com/advisories/mssql-udp.txt
Affects: Microsoft SQL Server 2000
SQL Server 2000 listens on UDP port 1434 to enable easy discovery by clients. The client will broadcast a single byte with a value of 0x02 to this port and any SQL servers will reply with connect information. If a SQL Server receives a packet on this port with a first byte of 0x04 then any remaining data is copied to a buffer when building the name of a registry key to open. If too much data is there a stack based buffer overflow occurs allowing the attacker access. This is the bug that the Slammer worm ended up exploiting on the 25 th January 2003.

Oracle

Oracle TNS Listener Multiple Buffer Overflows
Discoverer: Nishad Herath, Brock Tellier, 27 th June 2001
Advisory: http://cert.uni-stuttgart.de/archive/bugtraq/2001/06/msg00401.html
Affects: Oracle 8i
If an attacker supplied an overly long parameter to one of the TNS Listener commands a stack based buffer was overflowed allowing an attacker to overwrite the saved return address and gain control of the Listener's path of execution. This allowed an attacker to run code in the security context of the account running the server, typically oracle on *nix and SYSTEM on Windows.

Oracle TNS Listener Long SERVICE_NAME Buffer Overflow
Discoverer: David Litchfield, 12 th June 2002
Advisory: http://www.ngssoftware.com/advisories/oratns.txt
Affects: Oracle 9i
If an attacker supplied an overly long service_name to the TNS Listener a stack based buffer was overflowed when logging the request allowing an attacker to overwrite the saved return address and gain control of the Listener's path of execution. This allowed an attacker to run code in the security context of the account running the server, typically oracle on *nix and SYSTEM on Windows.

Oracle PL/SQL External Procedures Arbitrary Command Execution
Discoverer: David Litchfield, 6th February 2002
Advisory: http://www.nextgenss.com/advisories/oraplsextproc.txt
Affects: Oracle 9i, 9, 8i and 8
Details: From within the database, if a PL/SQL procedure calls an external routine the database connects to the TNS Listener which then launches extproc. Extproc loads the external routine and executes it on behalf of the database passing the results back. None of the inter-process communication is authenticated and it is possible for a remote attacker, without a user ID and password, to connect to the listener and

pretend to be the oracle process. When exploited, an attacker can force extproc to load an arbitrary library, execute an arbitrary function, passing it arbitrary parameters. A real world attack would typically seek to load msvcr7.dll or libc and execute the system() function. The command would run with the privileges of the account running the listener. This is SYSTEM on Windows, by default, and oracle on *nix.

N.B. 1) Oracle never fixed this for version 8.1.7.4 – it is still vulnerable.

N.B. 2) On “patched” systems, attacks can still be effected locally without an Oracle user ID and password – or remotely (with a userID and password) using UTL_TCP from the database.

Oracle RDBMS Long Username Buffer Overflow

Discoverer: Mark Litchfield, 16th February 2003

Advisory: <http://www.nextgenss.com/advisories/ora-unauthrm.txt>

Affects: Oracle 9i, 9, 8i, 8

If an attacker supplied an overly long username when connecting to an Oracle database server, a stack based buffer was overflowed allowing an attacker to overwrite the saved return address and gain control of the program's path of execution. This allowed an attacker to run code in the security context of the account running the server, typically oracle on *nix and SYSTEM on Windows.

Oracle XDB Long FTP Password Buffer Overflow

Discoverer: David Litchfield, 10th July 2003

Advisory: <http://www.nextgenss.com/papers/exploitvariation.pdf>

Affects: Oracle 9i

If an attacker supplied an overly long password when connecting to an Oracle database server running the XDB ftp server, a stack based buffer was overflowed allowing an attacker to overwrite the saved return address and gain control of the process' path of execution. This allowed an attacker to run code in the security context of the account running the server, typically oracle on *nix and SYSTEM on Windows.

Oracle XDB Long Username Buffer Overflow

Discoverer: David Litchfield, 10th July 2003

Advisory: <http://www.nextgenss.com/papers/exploitvariation.pdf>

Affects: Oracle 9i

If an attacker supplied an overly long username or password when connecting to an Oracle database server running the XDB HTTP server, a stack based buffer was overflowed allowing an attacker to overwrite the saved return address and gain control of the process' path of execution. This allowed an attacker to run code in the

security context of the account running the server, typically oracle on *nix and SYSTEM on Windows.

Oracle PL/SQL External Procedures Buffer Overflow 1

Discoverer: Chris Anley, 25th July 2003

Advisory: <http://www.nextgenss.com/advisories/ora-extproc.txt>

Affects: Oracle 10g, 9i, 9

When Oracle patched the arbitrary command execution flaw in extproc they added code that logged attempts from remote systems. They logged the name of the library the attacker is attempting to load but use an unsafe call to sprintf() to do this. By passing an overly long library name a stack based buffer is overflowed allowing the attacker to overwrite the saved return address on the stack. From here the attacker can run arbitrary code thus allowing the attacker to take control.

Oracle PL/SQL External Procedures Buffer Overflow 2

Discoverer: David Litchfield, 31 August 2004

Advisory: <http://www.ngssoftware.com/advisories/oracle23122004.txt>

Affects: Oracle 10g, 9i, 9

When Oracle patched the buffer overflow introduced by the patch for the arbitrary command execution flaw in extproc they added code to check the length of the library name before passing it to the sprintf() call. If too long then no logging would take place – if short enough the logging would take place. After performing the length check however, any environment variables that were present were expanded. So, if the string \$PATH were found to be in the library name, this would be expanded. Thus an attacker could effectively bypass the length check and still exploit the unsafe call to sprintf().