

# [UNIX] Multiple MySQL Vulnerabilities (COM\_TABLE\_DUMP, COM\_CHANGE\_USER, read\_rows, read\_one\_row)

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-12/0038.html>

---

*From:* [support@securiteam.com](mailto:support@securiteam.com)

*Date:* 12/12/02

From: [support@securiteam.com](mailto:support@securiteam.com)

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 12 Dec 2002 16:55:20 +0200

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

Beyond Security would like to welcome Tiscali World Online to our service provider team.

For more info on their service offering IP-Secure, please visit [http://www.worldonline.co.za/services/work\\_ip.asp](http://www.worldonline.co.za/services/work_ip.asp)

-----

Multiple MySQL Vulnerabilities (COM\_TABLE\_DUMP, COM\_CHANGE\_USER, read\_rows, read\_one\_row)

---

## SUMMARY

e-Matters have discovered two flaws within the MySQL server that can be used by any MySQL user to crash the server. Furthermore one of the flaws can be used to bypass the MySQL password check or to execute arbitrary code with the privileges of the user running MySQLd.

e-Matters have also discovered an arbitrary size heap overflow within the MySQL client library and another vulnerability that allows to write '\0' to any memory address. Both flaws could allow DOS attacks against or arbitrary code execution within anything linked against libmysqlclient.

## DETAILS

Vulnerable systems:

- \* MySQL version 3.23.53a and prior
- \* MySQL version 4.0.5a and prior

Immune systems:

\* MySQL version 3.23.54

While auditing the MySQL sourcetree e-Matters discovered several bugs within the MySQL client and server that are listed below:

#### +++ SERVER +++ COM\_TABLE\_DUMP – Signed Integer Vulnerability

When handling the COM\_TABLE\_DUMP package MySQL < 4.x takes two chars from the packet, casts them directly to unsigned integers and uses them as length parameters for memcpy. Obviously negative values within the chars will turn into very big unsigned numbers. Because this is a heap to heap copy operation and there is no memory allocating function within the SIGSEGV handler we strongly believe this bug can only be used for denial of service attacks. Depending on the packet MySQLd will directly crash or hang in an endless loop of segmentation faults. This was tested against Windows, Linux and FreeBSD systems.

#### +++ SERVER +++ COM\_CHANGE\_USER – Password Length Vulnerability

In February 2000 Robert van der Meulen discovered a flaw within the main password authentication system of MySQL: The MySQL challenge response algorithm creates an expected response with exactly the length of the response provided by the client. So if the client sends only a one char response MySQL will check only one byte. But this means it is possible to give the correct response with only 32 tries (because the character set is only 32 characters big). When this bug was fixed in 2000 the MySQL authors simply added a check in the server that the response must be 8 chars long. However they forgot to add this check to the COM\_CHANGE\_USER command, too. So it is still possible for an attacker with a valid MySQL-account to compromise the other accounts that are allowed to login from the same host. For a local user this means he can break into the MySQL root account and so compromise all databases. This is especially dangerous in a shared environment or if the root user is allowed to login from other hosts than localhost. While the attacker can supply a one byte response to break into the other accounts he can also send an oversized one. If the response is longer than 16 chars the internal created expected answer overflows a stack buffer. If the response is long enough it is possible to overwrite the saved instruction pointer with bytes that are generated by the random number generator of the password verification algorithm. While this sounds hard or impossible to exploit, we successfully exploited this bug on e-Matter's Linux machines. Due to the fact that MySQL restarts on crash you have unlimited tries. Because of the limited set of characters generated by the random number generator we strongly believe that this bug is not exploitable on Windows, because it is not possible to overwrite the instruction pointer with valid controllable addresses.

#### +++ CLIENT +++ libmysqlclient read\_rows Overflow

When the MySQL client library receives answer rows from the server it wants to copy the answers into another buffer. Therefore it loops through the returned fields and copies them to the other location. This is done without actually checking if the stored field sizes are within the destination buffer boundaries. Additionally there is also a terminating

'\0' added to the end of all fields without checking for enough space within the destination buffer. Due to the fact that this bug gets already triggered by a simple SELECT query anything that is linked against libMySQL is potentially vulnerable. Due to the nature of this bug it is trivial to use it as denial of service attack against the client applications (A negative field size will do the job). If it possible to use this overflow to execute code on the client system is different from application to application. It depends mainly on the fact if malloc() overflows are exploitable on that particular system and if the application allows enough control over the heap structure by triggering different execution paths.

+++ CLIENT +++ libmysqlclient read\_one\_row Byte Overwrites  
When the MySQL client library fetches one row from the MySQL server it loops through the fields to remember pointers to the field values. The field sizes are trusted and not checked against out of boundary conditions. After remembering the pointer the previous field gets zero terminated. A malformed packet can supply any field size and so overwrite some arbitrary memory address with a '\0'. An invalid address will of course crash the client. Because the address that is written to is arbitrary (maybe hard to supply because it must be supplied as delta) all clients that make use of fetching the answer row by row are most probably vulnerable to arbitrary code execution exploits.

Finally it must be mentioned that an attacker can of course use a combination of the described attacks to break into a system or to get access to privileges he normally does not own. f.e. it is possible for a local user to crash the server with the COM\_TABLE\_DUMP bug (if he cannot takeover the root account with the COM\_CHANGE\_USER bug) and then bind a fake server to the MySQL port 3306. And with a fake server he can exploit the libmysqlclient overflow. Another scenario would be an attacker that tries to exploit his favorite mod\_scripting language to takeover the web server by connecting to an external fake server...

#### Vendor Response:

- 03. December 2002 Vendor was contacted by email.
- 04. December 2002 Vendor informs me that bugs are fixed and that they started building new packages.
- 12. December 2002 Vendor has released MySQL 3.23.54 which fixes these vulnerabilities.

#### Recommendation:

We suggest anyone using MySQL to upgrade to a new or patched version as soon as possible.

#### ADDITIONAL INFORMATION

The original advisory can be downloaded from:

<<http://security.e-matters.de/advisories/042002.html>>  
<http://security.e-matters.de/advisories/042002.html>

Securiteam: [UNIX] Multiple MySQL Vulnerabilities (COM\_TABLE\_DUMP, COM\_CHANGE\_USER, read\_rows, read\_one

The information has been provided by <mailto:s.esser@e-matters.de> Stefan Esser of e-Matters.

=====

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](mailto:list-unsubscribe@securiteam.com)  
In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto: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.