Introduction
The CVE-2012-2122 vulnerability affects Oracle MySQL 5.1.x before 5.1.63, 5.5.x before 5.5.24, and 5.6.x before 5.6.6, and MariaDB 5.1.x before 5.1.62, 5.2.x before 5.2.12, 5.3.x before 5.3.6, and 5.5.x before 5.5.23. It must also be stated that no official versions were affected. Only systems running on Linux with glibc optimized with SSE are in danger.

Exploitability
The root of the problem lies in expecting that C library function memcmp() returns only values between -255 and 255. When a user enters a password, hash is calculated using the entered password and adding a random scrambled string to it. This in turn is compared to the stored hash using memcmp() function. If the function returns 0, access is granted. The memcmp() function is defined in a figure below. Short explanation of the function is that it compares the first n bytes of memory area s1 and memory area s2. It returns a value less than zero if s1<s2, value greater than zero if s1>s2 and zero if s1=s2. The vulnerability arises when the memcmp() function is optimized as it is in glibc SSE. The function is optimized to calculate the difference between first non-equal bytes. Now depending on the position of the first non-equal byte memcmp() might return a value (not between -255 and 255) that is a multiple of 256 and that value is incorrectly converted to 0 (The return value is collected in a char-type variable and only the last byte of memcmp() return value is used). The optimization of the memcmp() function is what limits the vulnerability to Linux systems with SSE optimized GNU C Library's.
Possible attack

The attack begins by knowing a username that is valid (“root” for most cases). Now the attacker uses any password he likes to exploit the vulnerability. Since the hash generating protocol uses a random string that is different each time a password is entered the position of the first non-zero byte changes and there is no need to use different passwords for each attempt. Memcmp() function is intended to return values between -255 and 255 based on the difference of the compared strings. The optimized version of the function also returns values that are not in this range and are multiples of 256 that can be incorrectly transformed to 0. Because of the random nature of the hash this bug on average emerges once every 256 tries. It takes well under a second for 256 tries so basically there is no security whatsoever.

Figure below describes a one line code that is sufficient to exploit the vulnerability.
$ for i in `seq 1 1000`; do mysql -u root --password=bad -h 127.0.0.1 2>/dev/null; done

code

Figure 3. Example code for attack

Fixing the vulnerability

The solution to the problem is as easy as the attack. One option is to restrict access to the local system by modifying the my.cnf file (change “blind-address” parameter to “127.0.0.1”). If an attacker cannot reach the system, he cannot attack.

Another option is changing “password.c” by adding a function that checks if memcmp() returned 0 or not before turning into char-type variable. That way the conversion error cannot even occur and the bug is fixed.

Used literature

https://github.com/twitter/mysql/blob/master/sql/password.c
https://bugzilla.redhat.com/show_bug.cgi?id=CVE-2012-2122
http://www.tutorialspoint.com/c_standard_library/c_function_memcmp.htm
http://thehackernews.com/2012/06/cve-2012-2122-serious-mysql.html
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-2122