MySQL Backup und Security

Praktische Hinweise um MySQL sicher unter Linux zu betreiben

Lenz Grimmer <lenz@mysql.com>

(SAGE)/GUUG-Treffen
Hamburg, Germany

10. Mai 2007

MySQL AB

Einführung

Dieser Vortrag gibt einen Überblick über die verfügbaren Sicherheitsmechanismen des MySQL Servers und wie sie mit Sicherheitsfunktionen unter Linux kombiniert und verbessert werden können.

Zusätzlich werden verschiedene Werkzeuge und Strategien für die Datensicherung eines MySQL-Servers unter Linux vorgestellt.

Inhalt

- Verbesserung der Server-Sicherheit
  - Auf der MySQL-Seite
  - Auf Linux OS-Ebene
- MySQL Backup-Methoden
  - Physikalischer vs. logischer Backup
  - OSS Werkzeuge zum Erstellen von Backups
  - Kommerzielle Backup-Lösungen

Verbesserung der MySQL-Sicherheit

- Absicherung eines MySQL-Server ist erster Arbeitsschritt nach der Installation
- Standardinstallation ist bereits relativ sicher, aber einige weitere Schritte sind notwendig
- Zusätlich zu den von MySQL angebotenen Sicherheitsfeatures sollten ergänzende Maßnahmen auf OS-Ebene getroffen werden

MySQL Server Post-Installation

- Wichtig: ein Passwort für den **root** Benutzer

```
$ mysql -u root mysql
mysql> SET PASSWORD FOR
  root@localhost=PASSWORD('new_password');
```

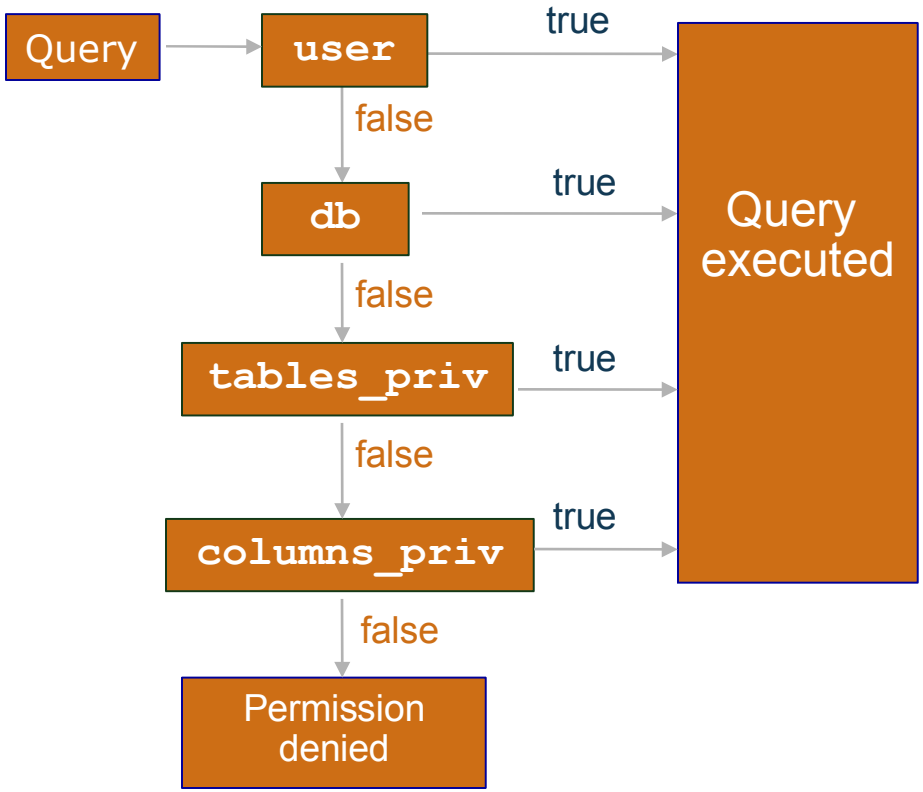
- Entfernen des **anonymous** Benutzers oder Vergabe eines Kennworts
- Entfernen der **test** Datenbank, wenn nicht benötigt
- All dies kann auf einen Schlag mit dem **mysql_secure_installation** Script erledigt werden

Zugangskontrolle

- **Verbindungsaufbau**
  - Wenn ein Client Verbindung aufnimmt, prüft der Server die **`user`** Tabelle ob er einen passenden Eintrag für **username**, **host** und **password** findet
- **SQL-Abfrage**
  - Für jede Abfrage prüft der Server die **`user`**, **`db`**, **`tables_priv`** und **`column_privs`** Tabellen

Zugangskontrolle bei Abfragen

Hat der Anwender die erforderlichen Rechte, die folgende Abfrage auszuführen?



Weitere Sicherheitshinweise

- Verwende die Option **`bind-address`** in **`my.cfg`** um den TCP-Port an eine bestimmte Netzwerkschnittstelle zu binden (z.B. **`127.0.0.1`**)
- Die Option **`skip-networking`** erlaubt nur Verbindungen über den lokalen Unix-Socket
- Zugriff sollte nur von bestimmten Hosts erlaubt sein
- Zugriff auf die **`mysql.user`** Tabelle sollte nur dem **`root`** Anwender erlaubt sein
- Lerne wie die **`SHOW GRANTS`**, **`SET PASSWORD`** und **`GRANT/REVOKE`** Kommandos verwendet werden
- Verwende phpMyAdmin oder MySQL Administrator für die Benutzerverwaltung

Weitere Sicherheitshinweise

- Beschränke **`PROCESS/SUPER/FILE`** Rechte auf das notwendige Minimum
- Speichere keine Klartext-Kennworte in Deiner Datenbank. Verwende stattdessen **`MD5()`**, **`SHA1()`** oder eine andere Einweg-Hashfunktion.
- Deaktiviere **`LOAD DATA LOCAL`** durch Setzen von **`local-infile=0`** in **`my.cnf`**
- Starte **`mysqld`** immer unter Verwendung eines nichtprivilegierten Benutzerkontos

Weitere Sicherheitshinweise

- Für Paranoiker:
  - Ersetze das **root** Konto mit einem anderen, schwerer zu ratenden Namen, um brute-force Wörterbuch-Attacken zu verhindern
  - Denke daran, die Historie des mysql Kommandozeilen-Clients (`~/.mysql_history`) zu löschen oder zu bereinigen, wenn Du damit Benutzerkontos oder Kennwörter editiert hast

Views und Stored Procedures

- **Views** können verwendet werden, um den Zugriff auch bstimmte Spalten oder Tabellen zu beschränken
- **Stored Procedures** können dazu verwendet werden, daß Tabellen nicht direkt vom Anwender oder einer Applikation modifziert werden
- Hinweis: verfügbar ab MySQL 5.0.x!

Verbessern der Zugriffsbeschränkung

- Reduziere die Zugriffsrechte auf das Datenbankverzeichnis mit **`chown`** und **`chmod`**
  - Anwender können Tabellen nicht zufällig **zerstören**
  - Anwender haben keinen Zugriff auf Daten, die sie nicht einsehen können sollen
- Auch die Logdateien sollten abgesichert sein:
  - Erneut könnten Anwender Einsicht auf Daten haben, die sie nicht haben sollten
  - Abfragen wie **`GRANT`** werden in den Logdateien gespeichert, jeder mit Zugriff auf die Logs könnte so an Passworte kommen
- Generell sollten reguläre Anwender keinen Shell-Zugriff auf dem DB-Server haben

Verminderung von Sicherheitsrisiken mit Linux

- Verwende iptables um den Server mit einer Firewall abzusichern
- Starte MySQL in einer `chroot()` Umgebung
- Verwende SELinux oder Novell AppArmor
- Starte den MySQL-Server in einer virtuellen Machine
  - Xen
  - UML (User Mode Linux)
  - VMware / Parallels
  - Virtuozzo

Absicherung der Daten und Kommunication

- Verschlüsselung des Netzverkehrs
  - OpenSSL
  - SSH tunnel
  - OpenVPN
  - Cipe
- Verschlüsselung des Datenverzeichnisses
  - cryptoloop devices
  - dm_crypt kernel module

Sicherung von MySQL-Daten

- Wann braucht man Backups?
- Was sollte gesichert werden?
- Wann sollten Backups durchgeführt werden?
- Wo sollen die Backups aufbewahrt werden?
- Wie können Backups durchgeführt werden?

Wann braucht man Backups?

- Hardware-Ausfall
  - Ein Teil der Daten einer Datenbank könnte durch einen Crash verloren gegangen sein
  - Ein Festplattenausfall führt mit großer Warscheinlichkeit zu starkem Datenverlust
- Anwenderfehler
  - Ein Anwender hat versehentlich eine DROP TABLE oder DELETE FROM Abfrage abgesetzt
  - Jemand (ein Administrator?) hat versucht, die Tabellen mit einem Texteditors zu bearbeiten, was üblicherweise zu Inkonsistenzen führt

Was sollte gesichert werden?

- Datenbank-Inhalte
  - Für vollständige Backups
  - Logischer oder Physikalischer Backup
- Log-Dateien
  - Für inkrementelle Backups
  - point-in-time recovery

Wann sollten Backups durchgeführt werden?

- Regelmäßig!
- Nicht zu den Stoßzeiten sondern nach Feierabend
- Statische Daten können weniger häufig gesichert werden

Wo sollten Backups aufbewahrt werden?

- Auf dem Datenbankserver
  - Aber wenigstens auf einem separaten Dateisystem/Volume oder zweiter Festplatte
- Kopiert zu einem anderen Server
  - Lokal oder in einem anderen Gebäude
- Backup auf Band/Platte
  - Lokal aufbewahrt oder woanders
- Nicht alles an einem Ort

Das Datenbankverzeichnis

- Standardmäßig werden alle Datenbanken und Logdateien im “data directory” gespeichert
- Eine Verzeichnis-Vorgabe ist in den Server einkompiliert
  - **`/usr/local/mysql/data/`** (tarball installation)
  - **`/var/lib/mysql`** (RPM-Pakete)
- Das Verzeichnis kann beim Start des Servers vorgegeben werden: **`--datadir=/your/path/`**
- Falls das Verzeichnis nicht bekannt ist, kann der Server danach befragt werden:
  - **`mysql> SHOW VARIABLES like 'data%';`**

Das Binary Log

- Enthält alle SQL-Anweisungen die tatsächlich Daten ändern
- Enthält weiterhin zusätzliche Informationen über jede Abfrage wie z.B. Zeitstempel
- Das Binary Log ist keine Textdatei, information ist in einem effizienteren Binärformat kodiert
- Inhalt kann mit **`mysqlbinlog`** eingesehen werden
- Aktivierung mittlels **`--log-bin[=file_name]`**
- Update-Logs werden sequenziell erzeugt z.B. **`file_name-bin.001`**, **`file_name-bin.002`**, etc.
- Das Binary Log ist transaktionskompatibel
- mysqld erzeugt eine Binary Log-Indexdatei, diese enthält die Namen aller verwendeter Binärlogs

Verwaltung des Binärlogs

- Aufgabe des Binärlogs
  - Leichtere Wiederherstellung nach einem Crash
  - Replikation
- **`SHOW MASTER LOGS`** zeigt alle Binärlogdateien auf dem Server
- **`FLUSH LOGS`** oder Server-Neustart erzeugen eine neue Logdatei
- **`RESET MASTER`** löscht alle Binärlogs
- **`PURGE MASTER`** löscht alle Binärlogs bis zu einem bestimmten Zeitpunkt

Das Error Log

- Wird der Server mittels **`mysqld_safe`** gestartet, werden alle **Fehlermeldungen** in das Error Log geleitet
- Die Datei enthält Informationen wann **`mysqld`** **gestarted** und **angehalten** wurde sowie alle **Fehlermeldungen** während des Betriebs

```
$ cat /var/log/mysql.err
000929 15:29:45  mysqld started
/usr/sbin/mysqld: ready for connections
000929 15:31:15  Aborted connection 1 to db: 'unconnected'
user: 'root' host: `localhost' (Got an error writing communication
    packets)
000929 15:31:15  /usr/local/mysql/bin/mysqld: Normal shutdown

000929 15:31:15  /usr/local/mysql/bin/mysqld: Shutdown Complete

000929 15:31:54  mysqld started
/usr/sbin/mysqld: ready for connections
```

mysqldump

- **mysqldump** speichert die Tabellenstrukturen und Inhalte als SQL-Anweisungen, die als Textdatei abgespeichert werden kann
  - **$ mysqldump mydb > mydb.20050925.sql**
- Man kann **individuelle Tabellen** oder **ganze Datenbanken** sichern
- The default output from **mysqldump** consists of **SQL statements**, **CREATE TABLE** statements for table structure and **INSERT** statements for the data
- **mysqldump** can also be used directly as input into another **mysqld** server (without creating any files)
  - **$ mysqldump --opt world | mysql -hwork.mysql.com world**

Recovering With Backups

Recovered database = Backup files + binary log

- In order to restore the tables to the state before a crash you will need both your **backup files** and the **binary log**
  - From the backup files you can restore the tables to the state they were at the **time of the backup**
  - From your **synchronised** binary logs you can extract the queries issued **between** the **backup** and **now**
- Beware, if you are recovering data lost due to **unwise** queries remember **not** to issue them again

Example SQL level restore

- Restore the last full backup

```
mysql < backup.sql
```

- apply all incremental changes done after the last full backup

```
mysqlbinlog hostname-bin.000001 |
mysql
```

MySQL table files backup

- Also called “physical” backup
- Database files can be simply be copied after issuing `FLUSH TABLES WITH READ LOCK;`
- The `mysqlhotcopy` Perl script automates this process (MyISAM table files only)
- Locking all tables for consistency can be expensive, if the file backup operation takes a long time

mysqlhotcopy

- **mysqlhotcopy** is a Perl script with which you can easily backup databases
- It can **only** be run on the **same** machine as where the databases are
- It does the following
  - **LOCK TABLES**
  - **FLUSH TABLES**
  - Copies the table files to the desired location with **cp** or **scp**
  - **UNLOCK TABLES**
- The user has to have **write access** to the target directory

Backing Up InnoDB Databases

- You can use the **`mysqldump --single transaction`** tool to make an on-line backup
- To take a '**binary**' backup, do the following:
  1. Shutdown the MySQL server
  2. Copy your **data** files, InnoDB **log** files, **`.frm`** files and **`my.cnf`** file(s) to a safe location
  3. Restart the server
- It is a **good** idea to backup with **`mysqldump`** also, since an error might occur in a binary file **without** you noticing it

OSS backup tools

- The usual suspects: **`cp`**, **`tar`**, **`cpio`**, **`gzip`**, **`zip`** called in a shell script via a **`cron`** job
- Use **`rsync`** or **`unison`** for bandwidth-friendly remote backups
- Complete network-based backup solutions like afbackup, Amanda or Bacula provide more sophisticated features (e.g. catalogs)

Linux backup support

- LVM snapshots
- DRBD (“RAID1 over the network”)
- Distributed file systems
  - OpenAFS
  - GFS
  - Lustre
  - Novell iFolder

Backup using LVM snapshots

- Linux LVM snapshots provide a very convenient and fast backup solution for backing up entire databases without disruption
- The snapshot volume does not need to be very large (10-15% are sufficient in a typical scenario)
- A backup of the files from the snapshot volume can be performed with any tool
- I/O performance may be degraded due to the additional LVM logging

Linux LVM snapshot creation

Basic principle:

```
mysql> FLUSH TABLES WITH READ LOCK
$ lvcreate -s --size=<size> --name=backup
<LV>
mysql> UNLOCK TABLES
$ mount /dev/<VG>/backup /mnt
$ tar czvf backup.tar.gz /mnt/*
$ umount /mnt
$ lvremove /dev/<VG>/backup
```

The mylvmbackup script

- A Perl script for quickly creating backups of MySQL server data files using LVM snapshots
- The LVM snapshot is mounted to a temporary directory and all data is backed up using the tar program
- Use of timestamped archive names allows you to run **`mylvmbackup`** many times without danger of rewriting old archives.
- requires Perl and the DBI and DBD::mysql modules
- Available from http://www.lenzg.org/mylvmbackup/

MySQL replication

- Backing up a replication slave is less time-critical (the master is not blocked for updates)
- A slave can use different storage engines
- One Master can replicate to many slaves
- Keep the limitations of MySQL replication in mind
- Make sure to back up the `master.info` and `relay-log.info` files as well as any `SQL_LOAD-*` files (if `LOAD DATA INFILE` is replicated)

Kommerzielle Backuplösungen

- Acronis True Image
- ARCServe
- Arkeia
- InnoDB HotBackup
- SEP sesam
- Veritas vxfs snapshots
- Zmanda

Backup Method Comparison

- The output from **`mysqldump`** is portable to any other DBMS (without the **`--opt`** option) whereas the copied files only work with MySQL
- The file copying methods are **much faster** than **`mysqldump`**
- So it comes down to **your** preferences:
  - Which **tool** do you prefer to use
  - Speed vs. portability

Backup Principles

- Perform backups regularly
- Turn on the binary update log
  - The update logs are needed to restore the database without losing any data
- Synchronise your update log files with your backup files
  - Use **`FLUSH LOGS`**
- Name your backups consistently and understandably
  - Include the date in the file name **`mydb.20050925.sql`**
- Store your backups on a different file system than where your databases are
- Backup your backup files with file system backups

General backup notes

- Putting the binary logs on a different file system (or even a different drive) than the data directory is recommended (increases performance and avoids data loss)
- Make sure the backup is consistent and complete!
- Define **backup schedules** and **policies** as well as **recovery procedures**
- **Test** that these actually work!

The MySQL Online Backup API

- Work is in in progress to define an API to perform a streaming MySQL online backup, independent of the Storage Engine
- Transactional tables will contain data only from committed transactions
- Non-transactional tables will contain data only from completed statements
- Referential integrity will be maintained between all tables backed up with a specific backup command
- The spec is now available for comments/review on MySQL Forge: http://forge.mysql.com/wiki/OnlineBackup

Thank you!

Questions, Comments?

Lenz Grimmer <lenz@mysql.com>