Sichere Installation und Verwendung von MySQL

Da MySQL ein serverbasiertes DBMS ist, stellt es besondere Anforderungen an die Sicherheit. Der folgende Artikel beschreibt die kritischsten Sicherheitslücken nach einer Standardinstallation von MySQL und wie man diese schließen kann.

Im MySQL Handbuch finden sich auch viele Hinweise zu dem Thema, jedoch nicht so verständlich erklärt wie in dem folgenden Artikel. Die Sicherheitshinweise im MySQL Handbuch finden Sie unter:
http://dev.mysql.com/doc/refman/5.1/de/security.html

Know-how: MySQL-Anwendungen sichern

Artikel aus Internet Professionell Ausgabe 9/2006
Autor: Max Bold
URL: http://www.testticker.de/ipro/praxis/security/article20060906020.aspx

Datenbank abschotten

In einer Lamp-Umgebung mit aufgesetzter Web-Anwendung gilt es zahlreiche sicherheitsrelevante Aspekte zu beachten. Der Artikel zeigt, wie Sie den MySQL aufsetzen und sicher konfigurieren, und wie Skripts ohne Risiko auf Datenbanken zugreifen können.

Darum geht es

Klassische Web-Anwendungen basieren auf einer Kombination von Webserver, Datenbankserver und der eigentlichen Anwendung, geschrieben in einer Skriptsprache wie PHP. In dieser Kette darf es in Sachen Sicherheit keinen Schwachpunkt geben. Dies gilt auch und insbesondere für die Datenbank-Komponente, die in der Mehrzahl der Fälle die Open-Source-Datenbank MySQL sein dürfte.

Eine Sicherheitsstrategie, die diese Komponente gegen Angriffe von außen absichern will, muss an verschiedenen Stellen ansetzen.

Die Security-Aspekte sind schon bei der Installation des Datenbank-Servers zu berücksichtigen. Wer hier den Status einer Standard-Installation beibehält, handelt fahrlässig. Auch am Webserver können einige Stellschrauben gedreht werden, die dessen Kommunikation mit dem Datenbank-Server sicherer macht.

Und schließlich kommt der Absicherung der eigentlichen Anwendung eine entscheidende Bedeutung zu, um zum Beispiel die Möglichkeit einer SQL-Injection weitgehend auszuschließen.

Vorhandene Infrastruktur

Zu berücksichtigen ist natürlich, auf welcher Infrastruktur eine solche Kombination betrieben wird. Davon hängt es weitgehend ab, welche Eingriffsmöglichkeiten bei den einzelnen Komponenten überhaupt zur Verfügung stehen.

Ein dedizierter Root-Server oder ein V-Server, wie sie von allen klassischen Providern angeboten werden und auf dem ein klassisches Lamp-System installiert ist, stellt den Benutzer natürlich vor die größten Herausforderungen. Hier trägt der Mieter für alle Komponenten die alleinige Sicherheitsverantwortung.

Demgegenüber sind die Eingriffsmöglichkeiten bei einem Managed Server oder gar bei einem Shared-Hosting-Paket wesentlich eingeschränkter. Hier konfiguriert der Provider die Umgebung und sorgt schon aus eigenem Interesse für einen optimalen Sicherheitslevel bei Web- und Datenbank-Server. Dadurch kann sich der Mieter und Anwender voll auf die Sicherheit seiner Anwendungen und Applikationen konzentrieren.

MySQL sicher installieren

Von Haus aus bietet der MySQL-Server eine Fülle unterschiedlicher Sicherheitsfeatures, die einem geschulten Administrator eine Vielzahl von Schutzmöglichkeiten für die sensiblen Daten zur Verfügung stellen.

Bereits bei der Installation des MySQL-Servers auf einem Linux-System sollte man die Augen offen halten. Das System legt nämlich diverse Benutzer automatisch an, darunter auch den User root mit uneingeschränkten Zugriffsrechten. Das Problem: Ein Passwort für diesen User wird nicht automatisch generiert. Eine der ersten Aufgaben nach der Installation ist daher das Anlegen eines solchen Passworts. Auf der Kommandoebene lässt sich dies wie folgt erledigen:

```
mysql> mysql -u root
mysql> SET PASSWORD = PASSWORD('meinkennwort');
```

Damit das neue Passwort nicht im Klartext in der Datenbank gespeichert ist, wird die Funktion PASSWORD zur Verschlüsselung verwendet. Alternativ können Sie dieses Passwort auch mit einem komfortablen Admin-Tool wie phpMyAdmin anlegen.

Ähnlich wie auf einem Linux-System besitzt der User root für den MySQL-Server uneingeschränkte Zugriffsrechte. Sie sollten diesen User daher nur verwenden, wenn Sie für administrative Aufgaben dessen Rechtespektrum auch benötigen.

In Web-Applikationen, die Zugriff auf eine Datenbank benötigen, mit dem User root zu operieren, wäre verantwortungslos. Sie sollten deshalb weitere Benutzer einrichten und deren Berechtigungen explizit auf einzelne Datenbanken oder Tabellen sowie den Zugriff von bestimmten IP-Adressen beschränken.

Benutzer einrichten

Dafür kann ein Blick auf das Zugriffssystem von MySQL hilfreich sein. Die Kontrolle über die MySQL-Datenbanken regeln in der Datenbank mysql die Tabellen:

- user: Hier werden Benutzer und Passwörter verwaltet und globale Berechtigungen für alle Datenbanken festgelegt.
- db: Benutzer, die uneingeschränkte Berechtigungen auf alle Tabellen einer Datenbank haben, werden in dieser Tabelle gepflegt.
- host: Basierend auf dem Hostnamen des Clients bestimmt diese Tabelle die Berechtigungen auf alle Tabellen einer Datenbank.
- tables_priv: Hiermit werden Benutzerrechte gezielt einzelnen Tabellen einer Datenbank zugeordnet.
- columns_priv: Hiermit werden spezifische Rechte für einzelne Spalten einer Tabelle bestimmt.

Die Zugangskontrolle erfolgt in MySQL in zwei Phasen. In der ersten Phase, wenn der Client eine Verbindung zum Server aufbaut, prüft MySQL anhand der Informationen der Tabelle user, ob eine Verbindung zulässig ist.

Hat ein erfolgreicher Verbindungsaufbau stattgefunden, wird für jeden Request des Clients zusätzlich geprüft, ob hierfür ausreichend Berechtigungen vorliegen. Dabei prüft MySQL die restlichen Systemtabellen der MySQL-Datenbank ab. Jede der fünf Tabellen hat dabei ihre eigene Daseinsberechtigung und gibt dem Administrator ein mächtiges Werkzeug an die Hand, um Client-Berechtigungen von globalen Datenbankrechten bis hin zu spaltenbezogenen Berechtigungen detailliert zu steuern.

Benutzerrechte zuordnen

Bei der Vergabe der Berechtigungen über die einzelnen Datenbanken und deren Tabellen helfen die Anweisungen GRANT und REVOKE. Der allgemeine Syntax der GRANT-Anweisung ist wie folgt:

GRANT berechtigungen [(spaltenname)] ON [datenbank.tabelle] TO benutzer [IDENTIFY BY 'passwort'] [WITH GRANT OPTION]

Gleichzeitig mit dem Anlegen des neuen Users in der Tabelle user setzt GRANT für diesen über IDENTIFY BY das Passwort, das in verschlüsselter Form gespeichert wird. Aus Sicherheitsgründen sollten keine Benutzer ohne Passwort verwendet werden. Zu beachten ist noch, dass Änderungen der Benutzertabellen nicht sofort durch MySQL erkannt werden. Erst nach Aufruf von

mysql>FLUSH PRIVILEGES

werden die neuen Rechte geladen. Sie sollten in jedem Fall der Versuchung widerstehen, den bequemen Weg über den MySQL-User root und womöglich noch ohne gesetztes Passwort bei der Implementierung einer Web-Applikation zu gehen. Bei der Vergabe von Berechtigungen sollte man eher restriktiv vorgehen und jedem User nur die Rechte im System einräumen, die er für die Ausführung seiner Aufgaben benötigt. So genügt es in den meisten Fällen, wenn der in einer Web-Applikation verwendete Benutzerzugang zum MySQL-Server lediglich über select, insert und update verfügt.

Neben den Benutzern des MySQL-Servers sollte man auf einem Linux-System auch die Berechtigungen im Auge behalten, mit denen der MySQL-Server-Prozess arbeitet und mit denen die Dateien im File-System abgelegt sind. Der MySQL-Server sollte niemals als root laufen, sondern über ein eigenes Benutzerkonto verfügen, das keine Administrationsrechte besitzt. Der Eintrag

```
[mysqld]
user=mysql
```

in der Datei /etc/my.cnf legt fest, dass der MySQL-Server als Benutzer mysql gestartet wird. Weiterhin empfiehlt sich ein Blick auf das MySQL-Datenverzeichnis. Hier legt MySQL den kompletten Inhalt aller Tabellen ab. Sämtliche Dateien sollten vor dem Zugriff von Unberechtigten geschützt sein, um so den Datendiebstahl zu verhindern. Der Befehl

ls -l /var/lib/mysql

zeigt unter Linux sämtliche MySQL-Datenbanken mit den zugehörigen Dateirechten an.

Netzwerkzugang deaktivieren

Sowohl die Administration als auch der Zugriff auf die gespeicherten Daten eines MySQL-Servers können lokal oder über das Netzwerk erfolgen. Die Kommunikation zwischen Server

und Client über das Netzwerk erfolgt standardmäßig über den TCP-Port 3306. Wer die Möglichkeit hat, den Datenverkehr zwischen Server und Client abzuhören, kann Daten ausspähen und Kenntnis über Logins und Datenstrukturen erlangen.

Sind MySQL-Server und Web-Applikation auf demselben Server installiert, so kann die Netzwerkverbindung des Servers komplett deaktiviert werden. Hierfür wird der MySQL-Server mit der zusätzlichen Option --skip-networking gestartet. Dazu wird unter Linux die Datei mysqld_safe editiert. Suchen Sie in dem File nach --skip-locking und fügen sie zusätzlich die Option --skip-networking ein. Mit dem nächsten Neustart steht die Netzwerkverbindung für den MySQL-Server nicht mehr zur Verfügung. Unter Linux kann die lokale Kommunikation mit dem Server dann nur noch über den Unix-Socket mysql.sock erfolgen.

Neben dem kompletten Abschalten der Netzwerkverbindungen bietet der MySQL-Server alternativ die Möglichkeit, den Zugriff aus dem Netzwerk auf bestimmte Rechner oder Subnetze zu beschränken. Für jeden Benutzer in der MySQL-Benutzerdatenbank muss daher angegeben werden, von welchen Rechnern aus er welche Berechtigungen besitzt. Bei der Administration der verschiedenen Hosts ist wiederum das MySQL-Administrator-Tool behilflich. Über das Kontextmenü zu einem Benutzer (rechte Maustaste über User drücken) lassen sich neue Rechner zu einem Benutzer hinzufügen. Als zusätzlichen Schutz empfiehlt sich der Einsatz einer Firewall, die sämtliche nicht erwünschten Verbindungen auf den Port des MySQL-Servers blockiert.

Sicherer Webserver

Ein sicher aufgesetzter MySQL-Server mit einem solide konfigurierten Zugriffsschema ist eine Sache. Tun sich jedoch Sicherheitslücken in der Webserver-Umgebung auf, kann dies auch gravierende Rückwirkungen auf die Sicherheit des Datenbank-Servers haben. Daher sollte man diesen Aspekt nicht aus den Augen verlieren, zumal mit kleinen Eingriffen schon eine große Wirkung erzielt werden kann. Es liegt zum Beispiel in der Natur der Sache, dass TCP-basierte Dienste wie zum Beispiel ein Webserver für den Zugriff aus dem Internet zugänglich sind. Klar ist auch, dass damit die Gefahr einer Hacker-Attacke gegeben ist. Ein Plus an Sicherheit kann man hier mit wenig Aufwand dadurch erzielen, dass man nur die Dienste freischaltet, die für den geplanten Einsatzzweck unbedingt notwendig sind.

Fungiert das Linux-System ausschließlich als Webserver und ist zur Pflege kein Remote-Zugriff zum Beispiel via FTP erforderlich, brauchen auch keine weitergehenden Dienste außer dem Web-server freigeschaltet werden.

Die von aktiven Diensten verwendeten Ports lassen sich im Rahmen der Firewall-Konfiguration einschränken. Die komplette Aktivierung und Deaktivierung von Diensten erfolgt in der Regel über den Runlevel-Editor.

Auch beim Webserver Apache unterschätzen viele Anwender die Gefahren und Schlupflöcher, die eine potenzielle Gefahrenquelle darstellen. Daher sollte man folgende Sicherheitstipps unbedingt beachten.

Es ist wichtig, den Server stets auf einem sicheren Release zu betreiben. Auch der Apache ist nicht frei von Bugs und Schwachstellen, an deren Behebung das Entwicklerteam ständig arbeitet. Man sollte sich als Apache-Betreiber daher ständig über den Entwicklungstand und eventuelle Patches informieren und insbesondere sicherheitsrelevante Updates sofort einspielen.

Rechte im Dateisystem

Im Normalfall bietet Apache seinen Dienst über den HTTP-Server-Standard-Port 80 an. Ein Prozess benötigt unter Linux Root-Berechtigungen, um sich an einen solchen Port zu binden. Nach dem Start ändert der Apache-Server allerdings seine Benutzer-ID und die Benutzergruppe für einen sicheren und ungefährlichen Dateizugriff. So finden sich in der Konfiguration in httpd.conf die Einträge

User nobody
Group nogroup

Als User nobody verfügt der Serverprozess kaum über Berechtigungen. Dieser Tatsache sollten Sie auch bei den Rechten für Dateien und Verzeichnisse unterhalb des Apache-Verzeichnisses Rechnung tragen. Ändern Sie die Berechtigungen für Konfigurationsdateien so, dass nur Sie darauf Schreib- und Lesezugriff haben.

PhpMyAdmin sichern

Eine weitere Gefahrenquelle im Umfeld von MySQL stellen Skripts dar, die man auf der eigenen Site installiert und die mit einer MySQL-Datenbank zusammenarbeiten.

Ein Paradebeispiel ist das beliebte Admin-Tool phpMyAdmin. Es ist eine optimale Web-Oberfläche zur komfortablen Pflege von MySQL-Datenbanken und Tabellen – und bei falscher Installation ein ebenso komfortables Einfallstor für Hacker. Eine absolut tödliche Konstellation stellt dabei ein MySQL-Server ohne gesetztes Root-Passwort und eine Standard-Installation von phpMyAdmin im Document-Root dar, quasi eine Prachtstraße für jeden Hacker ins Herz Ihres Datenbestandes.

Wo immer dies möglich ist, sollten Sie daher phpMyAdmin außerhalb des Document-Roots installieren oder – wo dies nicht möglich ist – zumindest innerhalb eines geschützten Verzeichnisses. Passwörter in der Config-Datei abzulegen ist nicht zu empfehlen, schon gar nicht das Root-Passwort. Es schadet auch nicht, die Existenz von phpMyAdmin durch Umbenennung des Verzeichnisses zu verschleiern. Es genügt in der Regel, wenn Sie wissen, dass sich Ihre phpMyAdmin-Installation im Verzeichnis /sonstiges/ befindet. Suchmaschinen kann man mit entsprechenden Einträgen in der Datei robots.txt die Recherche erschweren.

Sichere Skripts

Auch andere Skripts mit Datenbankzugriff sollte man unter dem Aspekt Security genau unter die Lupe nehmen. Dass der Zugriff auf den Datenbank-Server aus einem Skript heraus keinesfalls mit dem Root-Account erfolgen soll, wurde schon erwähnt. Es besteht in der Regel auch keine Notwendigkeit für so weit gehende Zugriffsrechte. Dies gilt nicht nur für eigene Programme, sondern auch für fertige Applikationen. Bei mehreren unterschiedlichen Applikationen kann es darüber hinaus nicht schaden, für jede einen eigenen MySQL-Account einzurichten. Damit reduziert man die Gefahr, dass beim Knacken eines Accounts dem Hacker sofort alle Zugänge offen stehen.

Vielfach werden zur Installation Skripts benötigt, die nach der Installation wieder gelöscht werden müssen, da sie in Händen eines böswilligen Users eine gefährliche Waffe darstellen. Oftmals können damit ohne Probleme MySQL-Zugangsdaten abgerufen werden. Damit ist das Tor für schwerwiegende Angriffe dann meilenweit offen. Man sollte daher mit solchen Tools sehr vorsichtig umgehen. Wenn sie nach der Installation nicht mehr benötigt werden, sind sie zu löschen. Ansonsten müssen Skripts dieser Art in jedem Fall vor dem Zugriff unberechtigter User geschützt werden.

Auch die Konfigurationsdateien von solchen Applikationen sollte man im Auge behalten. Darin stehen oftmals sensible Daten, die Hackern Zugriff auf sensible Bereiche freigeben.

SQL-Injection

SQL-Injection ist eine Methode, mit der ein Angreifer bösartigen Code in Abfragen einfügt, die in Ihrer Datenbank ausgeführt werden. Das läuft in etwa nach folgendem Schema ab:

```
<?php
$query = "SELECT login_id FROM users WHERE user='$user' AND pwd='$pw'";
mysql_query($query);
?>
```

Jeder kann sich als beliebiger Benutzer anmelden, indem er für die Abfrage einen String wie http://example.com/login.php?user=admin'%20OR%20 (user='&pwd=')%20OR%20user=' einsetzt, der effektiv die folgenden Anweisungen ausführt:

```
<?php
$query = "SELECT login_id FROM users WHERE user='admin' OR (user = '' AND pwd='') OR
user='"";
mysql_query($query);
?>
```

Noch einfacher ist der URL http://example.com/ login.php?user=admin'%23, der die Abfrage SELECT login_id FROM users WHERE user= 'admin' #' AND pwd='' ausführt. Beachten Sie, dass das Zeichen # in SQL einen Kommentar einleitet.

Es handelt sich wiederum um einen einfachen Angriff, dem Sie glücklicherweise genauso einfach vorbeugen können. Verwenden Sie dazu (in PHP) die Funktion addslashes(), die vor jedes einfache Anführungszeichen ('), jedes doppelte Anführungszeichen ("), jeden Rückwärtsschrägstrich (₩) und jedes NULL-Zeichen (₩0) einen Schrägstrich anfügt. Es gibt weitere Funktionen zum Absichern der Eingabe, zum Beispiel strip_tags().

Variablen prüfen

Das Haupteinfallstor für Hacker sind bekanntlich Formulare, deren Daten ungeprüft von einer Anwendung übernommen werden. Es ist daher ein absolutes Muss, jede Eingabe vor dem Weiterreichen an den Datenbank-Server auf Validität und Unschädlichkeit zu prüfen. Bei eigenen Programmen lässt sich diese Vorgabe bei sorgfältigem Arbeiten noch umsetzen. Schwieriger ist dies bei Open-Source-Applikationen wie etwa Foren, Weblogs oder Communitys, die sehr stark auf die Interaktivität mit den Besuchern und Nutzern bauen. Wer eine solche Applikation installiert, tut dies in der Regel im guten Glauben, dass sie sicher ist. Diese Einstellung kann jedoch trügerisch sein. Selbst etablierte Anwendungen wie phpBB sind nicht dagegen gefeit, dass immer wieder Sicherheitslücken entdeckt werden, die zum Beispiel SQL-Injection ermöglichen.

Die Chance liegt darin, dass sich so etwas vor allem bei beliebten Applikationen schnell herumspricht. Allerdings muss der Anwender dann umgehend reagieren und die angebotenen Sicherheits-Updates installieren.