Segurança com o MySQL

Anderson Pereira Ataides

1. Introdução

O MySQL sem dúvida nenhuma, é o banco de dados *open source* mais conhecido do mercado e provavelmente o mais utilizado. Ele é rápido, simples, funcional e hoje implementa recursos que o colocam próximo a grandes nomes como Oracle. Até pouco tempo um recurso extremamente útil e que era fator que impedia utilização em várias empresas, é o suporte à transação. Desde o lançamento da versão MAX, o MySQL já dá suporte a transações, derrubando este impecílio à sua utilização. Mais tarde, com a introdução das tabelas InnoDB, o MySQL ganhou mais um poderoso recurso: integridade referencial. Assim o MySQL passa a implementar os principais conceitos que aprendemos nos livros sobre banco de dados, fazendo-o ser uma alternativa viável para ser adotado no desenvolvimento de aplicativos.

Do ponto de vista do desenvolvedor de sistemas, o fato de ter implementado o suporte a transações e integridade referencial, já faz do MySQL um sistema seguro, já que a corrupção de dados é minimizada por estes dois recursos. Porém esta segurança não diz respeito a tentativas de acesso indesejado, mas apenas que as instruções SQL serão executadas de forma que o banco de dados permaneça sempre íntegro.

Pensando do lado do administrador do sistema, porém, o fato de o MySQL implementar os melhores recursos não o livra de situações que o sistema pode estar sujeito e das vulnerabilidades que devem ser sanadas com o intuito de não só manter a integridade dos dados mas também sua privacidade e sua própria existência. De que adianta o sistema implementar e o desenvolvedor usar transação se o sistema de arquivos está aberto para qualquer um copiar o repositório de dados? O que adianta o sistema se preocupar em fazer a remoção em cascata para preservar a integridade se qualquer um pode ter acesso ao banco de dados? Para alertar os usuários do MySQL sobre estes problemas, surgiu a idéia de escrever este texto.

Este texto trata exclusivamente do MySQL instalado em sistema Linux, apesar de o banco de dados poder ser instalado em outras plataformas. A escolha do Linux se deu por vários motivos dos quais podemos destacar pelo menos dois:

O Linux é livre

> Assim como o MySQL, o Linux é um software livre que pode ser instalado em qualquer equipamento sem preocupação com licenças de uso ou restrições legais.

O Linux implementa recursos sofisticados de segurança

> O Linux é extremamente flexível na implementação de políticas de segurança, podendo deixá-lo extremamente restritivo no acesso aos recursos da máquina, o que não acontece em outros sistemas (Windows 98 por exemplo).

Este trabalho apresenta aspectos fundamentais de segurança a ser observado na administração do banco de dados MySQL. A primeira parte refere-se aos recursos do próprio sistema operacional, como gerenciamento de usuários do sistema e permissões de acesso a arquivos. Na segunda parte, tratamos do sistema de privilégios internos do MySQL, ou seja, como ele valida usuários para acessar determinados recursos do banco de dados. O próximo assunto trata da utilização do MySQL em rede, onde e quando deve ser permi-

tido este tipo de conexão. Finalmente são feitas algumas considerações e principalmente recomendações para deixar o MySQL "quase" imune à ataques e acessos indesejados.

Espero que a leitura deste texto seja agradável e que venha a contribuir com o conhecimento sobre este pequeno e poderoso sistema de banco de dados. Contando com sua eficiência, facilidade de uso e implementando esquemas que o deixam seguro, o MySQL tem tudo para se tornar o que um concorrente proprietário prega sobre seu sistema: "imbatível"[1].

2. Segurança no sistema

Antes mesmo de pensar como um administrador de banco de dados (DBA), devemos pensar na segurança do sistema. Algumas considerações devem ser analisadas para evitar que, mesmo implementando controles de acessos a tabelas, banco de dados, o administrador seja surpreendido pela perda de dados pelo fato de alguém ter "acidentalmente" apagado o repositório do MySQL.

Apesar de implementar um sistema de validação robusto, o MySQL não tem como controlar acessos que deveriam ser bloqueados pelo sistema operacional. Acesso a arquivos, permissóes de usuários do sistema[2], ou mesmo do usuário sob o qual roda o servidor devem ser especialmente preparados para evitar que haja corrupção ou quebra da privacidade dos dados. Resumindo, apenas o banco de dados MySQL deve ter acesso à aos arquivos de dados do MySQL.

2.1. Sistema de arquivos

Como já foi falado anteriormente, apenas o banco de dados deveria ter acesso aos arquivos onde são armazenados os dados. No mundo Linux, esta restrição é bem simples de ser implementada, já que ele tem um esquema bem forte de autenticação que restringe o poder dos usuários do sistema. Apenas o usuário *root*[3] não pode ter o acesso bloqueado, já que ele pode redefinir as restrições estabelecidas.

Para evitar qualquer tipo de problemas, apenas o usuário sob o qual o servidor vai rodar, deve ter acesso ao diretório onde o MySQL guarda os arquivos de dados. Qualquer outro usuário deve ter o acesso a este diretório bloqueado tanto para leitura como para escrita. O acesso de escrita deve ser bloqueado por motivos óbvios: ninguém, a não ser o próprio banco de dados deve escrever nos arquivos de dados. Já a permissão de leitura merece uma explicação mais detalhada.

Ao bloquear o acesso à escrita nos arquivos do MySQL, garantimos que ninguém poderá fazer alterações nos arquivos, porém não conseguimos garantir o sigilo destes dados. Não é preciso nem decifrar o conteúdo dos arquivos para ler os dados ali armazenados, basta pegar os arquivos e, em outra máquina copiar os arquivos no diretório do MySQL e pronto. Você tem acesso a tudo que o outro banco de dados guardou. Para preservar o sigilo nos dados portanto, o acesso de leitura também deve ser bloqueado para os outros usuários que não o responsável pelo banco de dados.

2.2. Usuários do sistema

Para acessar o banco de dados, não é necessário criar uma conta no sistema operacional, pois o MySQL tem sua própria estrutura de validação desvinculando os usuários do banco de dados dos usuários do sistema. O único usuário do sistema que deve ter um tratamento especial é o que possui o processo servidor.

O MySQL, como qualquer processo no Linux possui um dono e conseqüentemente ele vai ter os mesmos privilégios que este dono tem no sistema. Assim a política adotada para este usuário é como em qualquer outra: a do menor privilégio. Se o servidor acessa só o banco de dados, por que dar poder ao dono do processo servidor para ler ou escrever arquivos externos ao banco de dados? Se o propósito do dono do banco de dados é apenas fazer o servidor funcionar, por que dar acesso de login a ele?

Existem algumas situações que o admistrador deve dar acesso ao sistema de arquivos para o dono do banco de dados, pois algumas funções no MySQL precisam deste acesso (LOAD DATA por exemplo). Se este for o caso, o admisitrador deve ter o cuidado para não deixar aberto acesso a arquivos importantes. É óbvio mas existem adminstradores que esquecem deste detalhe. Com um banco de dados preparado e um comando LOAD DATA é possível pegar por exemplo os dados do arquivo /etc/passwd para tentar obter as senhas dos usuários.

Uma observação importante é que o dono do banco de dados NUNCA deve ser o *root*. O motivo é óbvio: este usuário tem acesso a tudo, e acabamos de ver acima que este acesso indesejado é extremamente perigoso ao sistema. Um admisnitrador inexperiente poderia pensar que apenas o super-usuário do sistema pode ter acesso ao banco de dados e assim ele estaria seguro, mas o que acontece é que do outro lado, existe alguém desconhecido que vai enviar uma consulta ao banco de dados que está funcionando como super-usuário e que pode fazer qualquer coisa. O ideal portanto é que exista um usuário apenas para o banco de dados, sem direito a fazer mais nada que não seja manipular os dados do próprio banco de dados.

Aplicando estes procedimentos no servidor, estaremos garantindo a integridade do sistema, pois o usuário dono do servidor MySQL não tem privilégios. Também garantimos o sigilo dos dados, pois nenhum outro usuário vai ter acesso aos arquivos do banco de dados. Assim estamos implementando segurança no MySQL antes mesmo de fazer o servidor MySQL funcionar.

3. Sistema de autenticação do MySQL

O MySQL implementa um sistema de autenticação bastante robusto que é realizado em dois estágios. O primeiro verifica se o usuário pode conectar ao banco de dados e o segundo verifica se o usuário tem privilégios para realizar operações no banco de dados. O segundo estágio, portanto, é verificado a cada operação realizada pelo usuário.

Este sistema de privilégios é armazenado usando a própria estrutura do sistema em um banco de dados especial chamado *mysql*. Pela natureza dos dados que este banco de dados armazena, ele deve ter o acesso permitido apenas para o usuário *root* do MySQL. Os usuários comuns não necessitam de acessar este banco de dados, principalmente a tabela user, onde estão armazenadas as senhas dos usuários.

Para aceitar a conexão de um usuário, o MySQL não considera apenas o próprio usuário, mas também a máquina de onde o usuário está conectando. Dessa forma, você pode permitir o acesso de um determinado usuário somente de algumas máquinas específicas, bloqueando seu acesso de outros *hosts* que podem não ser confiáveis.

Existem duas maneiras de conceder privilégios aos usuários: usando os comandos GRANT e REVOKE, ou alterando diretamente as tabelas do banco de dados *mysql*. A melhor escolha é usar os comandos GRANT/REVOKE, pois o MySQL já altera as tabelas automaticamente, não sendo necessário entender em detalhes o significado de cada tabela e suas respectivas colunas. Se você alterar os privilégios manual-

mente além do risco de manipular dados de forma errada, você pode se esquecer de executar o comando FLUSH PRIVILEGES para tornar as alterações ativas.

Ao criar um novo banco de dados, deixe que apenas o administrador do banco de dados tenha acesso completo. Aos usuários comuns permita apenas acesso aos dados, evitando o acesso à estrutura do banco de dados. Assim um usuário comum com acesso "completo" deveria ter acesso aos comandos INSERT, DELETE, UPDATE e SELECT. Apenas o adminstrador do banco de dados deve ter acesso a comandos como DROP, CREATE ou ALTER. Dessa forma você está permitindo a cada um apenas o que ele necessita para o processamento de dados.

Exemplificando, vamos definir um certo "Alfredo" como administrador do banco de dados "expedicao" que vai ter como usuários um tal "Luciano" que, por ser desenvolvedor, pode alterar a estrutura do banco de dados e o "Thiago" que é o usuário final, ou seja, ele apenas precisa manipular os dados armazenados. Para definir estes três usuários, basta executar a seguinte seqüência de comandos SQL:

```
> GRANT ALL PRIVILEGES ON expedicao.* TO Alfredo@localhost
  IDENTIFIED BY 'senha_do_alfredo';

> GRANT SELECT,INSERT,UPDATE,DELETE,DROP,ALTER ON expedicao.*
  TO Luciano@localhost IDENTIFIED BY 'senha_do_luciano';

> GRANT SELECT,INSERT,UPDATE,DELETE ON expedicao.* TO Thiago@localhost
  IDENTIFIED BY 'senha_do_thiado';
```

Note que no exemplo acima, todos os usuários cadastrados têm acesso ao banco de dados apenas se estiverem conectando da máquina local, ou seja diretamente na máquina onde o servidor MySQL está rodando. Para permitir acesso de outros *hosts* basta repetir a consulta para um usuário alterando *localhost* para o nome ou endereço IP da máquina de onde será permitido ao usuário conectar ao banco de dados. Você poderia ainda usar o curinga '*%*' indicando que o usuário pode se conedtar de qualquer *host*, mas isto não é recomendado, pois a priori você não deve confiar em máquinas às quais você não tem informações.

É muito importante que o adminsitrador entenda pelo menos basicamente o funcionamento do sistema de privilégios do MySQL para evitar conceder a um usuário mais poder do que ele necessita. São vários tipos de privilégios que um determinado usuário pode ter além de SELECT,INSERT,UPDATE,DELETE,DROP e ALTER mostrados acima. É altamente recomendado fazer uma leitura no manual do MySQL para ver os privilégios disponíveis e como utilizá-los de forma correta.

4. Conexão via rede

Ao conectar ao servidor MySQL localmente, tendo um sistema bem configurado o MySQL já pode ser considerado bem seguro. Ao disponibilizar o acesso via rede, porém, criamos mais um ponto de vulnerabilidade deixando o sistema à mercê de ataques dos mais variados tipos. O simples fato de deixar uma porta aberta já aguça a curiosidade de certos usuários para tentar usar esta porta aberta como entrada para derrubar serviços ou outras formas de "atrapalhar" o funcionamento do sistema.

Em sua instalação padrão, o MySQL inicia permitindo conexões locais e conexões via rede. Na seção anterior, vimos como permitir que um usuário se conecte a partir de um *host*. O simples fato de não permitir a conexão de um usuário não siginifica que não teremos mais problemas porque a porta continua aberta para a rede. Pensando nesta "porta aberta", é necessário implementar mecanismos para que os dados que trafegam por esta porta não sejam lidos por alguém que não o servidor e o cliente.

Para ajudar a decidir como "esconder" os dados de *crackers*, devemos ter em mente como será desenvolvido o aplicativo que vai usar o banco de dados. Se for um ambiente web, onde o servidor web e o MySQL estejam na mesma máquina, não há motivos para liberar o acesso via rede. Neste caso o servidor deve ser iniciado com a opção *--skip-networking* que faz com que o MySQL funcione apenas com conexões locais via *sockets*. Se o aplicativo estiver em uma máquina e o servidor em outra como em ambientes cliente/servidor, ou mesmo web onde o servidor web está em uma máquina e o servidor MySQL em outra, esta opção não pode ser utilizada.

Nos casos onde o acesso a rede deve ser necessário, a primeira providência a ser tomada é permitir conexões aos usuários apenas das máquinas de onde eles têm permissão para acessar o banco de dados. Isto deve ser feito através dos comandos GRANT e REVOKE vistos anteriormente.

A segunda providência é estabelecer uma conexão segura com o servidor. A senha no momento da autenticação não é transmitida em texto plano, porém o algoritmo de criptografia não é forte e pode ser facilmente quebrado. Outro problema com a conexão estabelecida entre cliente e servidor é que todos os dados (requisições SQL e retorno dos dados) trafegam em texto plano e qualquer um rodando um *sniffer* pode ver o diálogo entre o servidor e o cliente.

Existem pelo menos duas soluções para este problema. A partir da versão 4.0.0, o MySQL tem suporte a SSL, que é um protocolo que utiliza diferentes algoritmos de criptografia para assegurar que os dados recebidos por uma rede pública são confiáveis. Outra solução é criar uma VPN usando aplicativos como SSH que criam um túnel criptográfico entre dois *hosts* e o *host* remoto passa a enxergar o servidor como se estivesse rodando localmente.

Usando o MySQL com suporte a SSL, você pode, ao criar um usuário, informar ao servidor que este usuário precisa ser autenticado usando também atributos do SSL além dos dados padrão (usuário, senha, nome/IP do *host*). Basta para isto acrescentar a cláusula REQUIRE no comando GRANT. Exemplificando, vamos fazer com que a autenticação do usuário Alfredo seja feita com SSL.

```
> GRANT ALL PRIVILEGES ON expedicao.* TO Alfredo@localhost IDENTIFIED BY
  'senha_do_alfredo' REQUIRE SSL;
```

O manual do MySQL dá todas as informações passo a passo para criar certificados para o MySQL e como configurar o MySQL para utilizar acesso seguro via SSL. Recomendamos a sua leitura para implementar este recurso.

Se o servidor MySQL deve ser visto apenas na rede local, o acesso externo deve ser bloqueado. Você pode fazer isto com o próprio esquema de privilégios do MySQL, mas assim apenas o acesso ao banco de dados estará restrito e a porta continuará aberta para a rede externa. A melhor alternativa para evitar este acesso indesejado é com a implementação de um *firewall*. Se a rede externa não deve acessar o MySQL o próprio *firewall* se encarrega de filtrar o acesso. Dessa forma a porta de acesso ao MySQL será fechada para conexões externas.

5. Dicas para deixar o MySQL "Imbatível"

Existem várias recomendações para que você deixe o MySQL o mais seguro possível. Vários problemas de segurança existem não por falha do sistema, mas do próprio administrador do sistema que deixa portas abertas permitindo ataques que podem (e devem) ser evitados com medidas simples. Aqui vai um checklist a ser observado em qualquer sistema rodando o MySQL. Este checklist é um resumo do que está no próprio manual do MySQL.

- Nunca conceda acesso a alguém à tabela *user* do banco de dados *mysql*. Com este acesso é possível obter as senhas de outros usuários do MySQL.
- Não mantenha senhas em texto plano em seus bancos de dados. Ao usar senhas, use funções de hash como MD5() ou SHA1().
- Aprenda e use o sistema de privilégios do MySQL. Dessa forma você evitará surpresas desagradáveis como um DROP TABLE de um usuário que não deveria ter este privilégio.
- Se o banco de dados for aberto à Internet, use criptografia. Use um protocolo como SSL ou SSH para estabelecer uma conexão segura entre o cliente e o servidor.
- Utilize senha para todos os usuários do MySQL. Um *cracker* geralmente usa um usuário cadastrado no sistema para fazer seus ataques.
- Nunca inicie o servidor com o usuário *root* do sistema operacional. Como este usuário tem acesso a tudo, alguém pode tentar utilizar deste privilégio para danificar o seu sistema.
- Não confie nas instruções SQL enviadas pelo cliente. Em ambientes web principalmente, faça uso de funções que "alteram" seqüências de caracteres que significam algum tipo de "código" de escape para comandos html ou do próprio SQL.
- Não conceda acesso a usuários de qualquer *host*. Uma máquina "externa" pode estar sendo monitorada (por um *sniffer*) ou pode não ser realmente a máquina de um usuário conhecido, mas de um *cracker* querendo acesso ao seu servidor.
- Limite recursos do usuário como número de conexões por exemplo. Estas limitações de recursos podem evitar ataques tipo DOS[4].
- Invista em um bom *firewall*. Com esta medida você evita tentativa ataques externos à sua rede local.
- Mantenha-se informado. Os ataques bem sucedidos ocorrem porque o invasor sabia de uma brecha de segurança que o administrador não conhecia ou ignorou no momento da configuração do servidor.
- Não libere nenhum recurso que não será utilizado. Quanto menos portas abertas, menor as chances de sofrer ataques.

6. Conclusões

O MySQL pode ser tão seguro quanto queira o seu adminsitrador. São inúmeras variáveis que devem ser levadas em conta ao configurar um sistema para funcionar de forma que os dados estejam o mais seguros possíveis. Além disto, soma-se a dificuldade de que a comodidade do usuário deve sofrer o mínimo de impacto sob o risco de deixar o sistema inutilizável.

O bom administrador vai conseguir este balanceamento se conhecer muito bem o sistema onde o MySQL vai funcionar. Ele não precisa saber programação, mas deve saber pelo menos como os aplicativos vão acessar o banco de dados de modo que ele possa "ajustar" o sistema para deixar disponíveis apenas os recursos necessários para o bom funcionamento destes aplicativos.

A palavra chave para manter o sistema seguro é *INFORMAÇÃO*. O administrador deve sempre visitar *sites* que tratam de segurança não só do MySQL, mas do sistema como um todo, pois o MySQL pode parar de responder aos clientes não por falha própria, mas do sistema onde ele está rodando. Uma outra ótima forma de manter-se atualizado sobre o MySQL é participando de listas de discussões onde existe toda uma comunidade trocando informações e experiências no uso do MySQL.

Uma outra grande fonte de informação é o próprio manual do MySQL. Muitas falhas poderiam ser evitadas se os administradores lessem e aplicassem o que está escrito no manual. O próprio *site* do MySQL (http://www.mysql.com/) tem muitos artigos que tratam não só de segurança, mas também de vários aspectos sobre o funcionamento do MySQL.

Finalmente, se em caso de problemas nenhuma fonte de informação conseguir ajudar, o adminsitrador deve ter humildade suficiente para admitir que não sabe tudo e procurar ajuda. Mesmo porque como ficou claro neste texto, as falhas podem vir de outro lugar e o admisitrador pode não perceber este ponto de vulnerabilidade. Basta pensar que a troca de idéias é importantíssimo para nossa evolução, afinal é isto que nos diferencia de outros seres: nossa capacidade de aprender e ensinar.

Referências

Kristy Westphal, *Secure MySQL Database Design*.

MySQL AB, *MySQL Reference Manual*.

Site oficial do MySQL: http://www.mysql.com/.

Notas

1. Imbatível é o adjetivo que a Oracle (http://www.oracle.com/) deu a seu banco de dados referindo-se à impossibilidade de se quebrar seus recursos de segurança.
2. Os usuários do banco de dados não precisam necessariamente ser usuários do sistema operacional, ou seja, o sistema de autenticação do MySQL é completamente independente do sistema operacional. O aplicativo de linha de comando mysql, por exemplo, usa o usuário do sistema apenas por conveniência, o que pode ser mudado pela opção -u.
3. *root* é o usuário que tem poder para fazer qualquer coisa no Linux, ou seja, é o super-usuário do sistema.
4. DOS = Denial Of Service (Negação de Serviço). Este tipo de ataque é realizado enviando várias requisições simultâneas a um servidor até que ele sobrecarregue e passe a não aceitar mais nenhuma conexão.