AUDITORIA EM SISTEMAS E SEGURANÇA DA INFORMAÇÃO

Professor Me. Vladimir Geraseev Junior

SEGURANÇA EM BANCO DE DADOS

SEGURANÇA EM BANCO DE DADOS

- Aspectos Gerais de segurança;
- Evitar violação de consistência dos dados;
- Segurança de acesso(usuários e aplicações);
- Segurança contra falhas(recovery);
- Manutenção de histórico de atualizações (Log) e backups do BD;

SEGURANÇA EM BANCO DE DADOS

- Segurança com Banco de dados livres (MariaDB);
- Consequencias de não ter um ambiente seguro;
- Recomendações para um ambiente seguro;
- Referências.

ASPECTOS GERAIS DE SEGURANÇA

Porque devemos ter segurança em um Banco de Dados?

Possuir informação hoje é ganhar agilidade, competitividade, previsibilidade, dinamismo, portanto, possuir informação é o mesmo que possuir um diferencial, uma vantagem competitiva.

Uma informação útil pode ser usada a favor ou contra você ou sua empresa. Por isso é importante que se possua informações corretas e uma forma eficiente de protegê-las, já que se algum dado crítico for alterado, destruído, ou divulgado sem autorização pode acarretar em prejuízos tanto para a empresa ou instituição, como para seus clientes ou usuários.

ASPECTOS GERAIS DE SEGURANÇA

De acordo com SÊMOLA (2003, p.12), os principais motivos para se proteger uma informação são :

o seu valor,

o impacto de sua ausência,

o impacto resultante de seu uso por terceiros,

a importância de sua existência e a relação de dependência com a sua atividade, e

a informação deve ser protegida em todo o seu ciclo de vida, desde sua criação, manuseio, armazenamento transporte e descarte.

Os SGBDs para garantir a segurança das informações,

devem possuir **controles de redundancia**, **concorrencia** e a capacidade de manter os dados integros, aplicando as **restrições de integridade.**

CONTROLE DE REDUNDÂNCIA

- A redundância é caracterizada pela presença de um elemento de informação duplicado.
- Sistemas de banco de dados devem ter capacidade de garantir que os dados não sejam redundantes.
- Esse controle é usualmente conhecido como integridade referencial.
- O controle de redundância não permite incluir dois registros com a mesma chave primária e excluir um registro que possua relacionamento com outras tabelas (chave estrangeira).
- Com isto, o controle de redundância evita a inconsistência de dados.
- Este padrão de integridade é o fundamento do modelo relacional, por isso é necessário que o banco de dados tenha a capacidade de gerenciar o controle de redundância.

CONTROLE DE CONCORRÊNCIA

- É um esquema usado para garantir que as transações são executadas de forma segura.

Uma das qualidades dos sistemas desenvolvidos é a multiprogramação que permite a execução de diversas transações visando o compartilhamento do processador. Nesse ambiente multiprogramado diversas transações podem executar concorrentemente. Por isso os sistemas precisam controlar a interação entre transações concorrentes com o objetivo de

- prevenir a violação da consistência do banco de dados.

Esse controle é feito por um conjunto de mecanismos definidos como esquemas de

- controle de concorrência.

Quando as transações são executadas concorrentemente a consistência do banco pode ser violada mesmo que cada transação individual esteja correta.

- Cabe ressaltar que nesse ambiente é importante o conceito da seriabilidade. É necessário que qualquer escalonamento produzido ao se processar um conjunto de transações concorrentemente seja computacionalmente equivalente a um escalonamento produzindo executando essas transações serialmente em alguma ordem.
- Diz-se que um sistema que garante esta propriedade assegura a seriabilidade.

CONTROLE DE CONCORRÊNCIA

- Os escalonamentos podem ser seriais e não-seriais.
- Escalonamentos seriais consistem de uma seqüência de instruções de várias transações onde as instruções pertencentes a uma única transação aparecem juntas.
- No escalonamento não serial as operações de uma transação são executadas intercaladas com operações de outra transação.
- Os mecanismos de controle de concorrência são classificados em mecanismos otimistas e pessimistas.
- Os mecanismos de controle de concorrência pessimistas são aqueles que buscam impedir antecipadamente os tipos de acesso concorrente a dados que podem gerar inconsistências. Isso pode ser feito bloqueando temporariamente o acesso de dados a algumas aplicações enquanto outra está acessando.
- Os mecanismos de controle de concorrência otimistas, ao invés de tentar evitar antecipadamente acessos inconsistentes permitem o livre acesso.
- No final da execução das aplicações é iniciado um processo que examina a incidência de inconsistência nos dados graças ao acesso concorrente.

RESTRIÇÃO DE INTEGRIDADE

- As restrições de integridade oferecem meios de assegurar que mudanças feitas no banco de dados por usuários autorizados não resultem em perda de consistência dos dados.
- As restrições de integridade podem resguardar o banco de dados contra danos acidentais.
- Uma restrição de integridade pode ser um predicado arbitrário que reaplica ao banco de dados.
- No entanto, os predicados arbitrários podem ser custosos para testes. Dessa forma é preciso limitar-se em restrições de integridade que possam ser testadas com um mínimo custo adicional.
- A forma mais elementar de restrição de integridade é a restrição de domínio.
- Em um sistema é possível que diversos atributos tenham um mesmo domínio.
- A definição de restrição de domínio não somente permite testar os valores inseridos no banco de dados como possibilita testar as consultas assegurando que as comparações façam sentido.

RESTRIÇÃO DE INTEGRIDADE

- A integridade referencial é usada para garantir que um valor que aparece em uma relação para um dado conjunto de atributos também apareça para um certo conjunto de atributos em outra relação.
- No modelo E-R (Entidade – Relacionamento), o esquema do banco de dados relacional derivado de diagramas E-R resulta em um conjunto de relacionamentos que possui regras de integridade referencial.
- Outro ponto de origem de restrições de integridade referencial são os conjuntos de entidades fracas. Isto porque o esquema relacional para cada conjunto de entidades fracas inclui uma chave estrangeira que leva a uma restrição de integridade referencial.
- As modificações em banco de dados podem violar as regras de integridade referencial.

EVITAR VIOLAÇÃO DE CONSISTÊNCIA DOS DADOS

Para evitar a violação dos dados e garantir a consistência, confiabilidade, podemos adotar algums mecanismos de segurança entre esses mecanismos podemos destacar:

Mecanismos de controles fisicos

- portas / trancas / paredes / blindagem /etc...

Mecanismos de controles lógicos

- Mecanismos de criptografia
- Assinatura digital
- Mecanismos de garantia da integridade da informação
- Mecanismos de controle de acesso e etc...

Dentre os mecanismos de controle lógico, vamos destacar a criptografia.

CRIPTOGRAFIA

A Criptografia é a técnica pela qual a informação pode ser transformada da sua forma original para outra ilegível, de forma que possa ser conhecida apenas por seu destinatário, o que a torna difícil de ser lida por alguém não autorizado. Assim sendo, só o receptor da mensagem pode ler a informação com facilidade.

A CRIPTOGRAFIA TEM QUATRO OBJETIVOS PRINCIPAIS :

- **confidencialidade da mensagem**: só o destinatário autorizado deve ser capaz de extrair o conteúdo da mensagem da sua forma cifrada. Além disso, a obtenção de informação sobre o conteúdo da mensagem (como uma distribuição estatística de certos caracteres) não deve ser possível, uma vez que, se o for, torna mais fácil a análise criptográfica.
- **integridade da mensagem**: o destinatário deverá ser capaz de determinar se a mensagem foi alterada durante a transmissão.
- **autenticação do remetente**: o destinatário deverá ser capaz de identificar o remetente e verificar que foi mesmo ele quem enviou a mensagem.
- **não-repúdio ou irretratabilidade do emissor**: não deverá ser possível ao emissor negar a autoria da mensagem.

CRIPTOGRAFIA

- Nem todos os sistemas ou algoritmos criptográficos são utilizados para atingir todos os objetivos listados acima.
- Normalmente, existem algoritmos específicos para cada uma destas funções.
- Mesmo em sistemas criptográficos bem concebidos, bem implementados e usados adequadamente, alguns dos objetivos acima não são práticos (ou mesmo desejáveis) em algumas circunstâncias. Por exemplo, o remetente de uma mensagem pode querer permanecer anônimo, ou o sistema pode destinar-se a um ambiente com recursos computacionais limitados.
- Tipos de Criptografia, MD2, MD4, SHA, Hash, MD5, MD6 (não utilizavél).
- Os mais indicados e comuns são o MD5, o SHA e o Hash

CRIPTOGRAFIA

- MD5 (**Message-Digest algorithm 5**) é um algoritmo de hash de 128 bits unidirecional desenvolvido pela **RSA Data Security, Inc**., descrito na **RFC 1321**, e muito utilizado por softwares com protocolo ponto-a-ponto (**P2P, ou Peer-to-Peer, em inglês**) na verificação de integridade de arquivos e logins.
- Foi desenvolvido em 1991 por **Ronald Rivest** para suceder ao **MD4** que tinha alguns problemas de segurança. Por ser um algoritmo unidirecional, uma hash **md5** não pode ser transformada novamente no texto que lhe deu origem. O método de verificação é, então, feito pela comparação das duas hash (uma da mensagem original confiável e outra da mensagem recebida).
- O **MD5** também é usado para verificar a integridade de um arquivo através, por exemplo, do programa md5sum, que cria a hash de um arquivo. Isto pode-se tornar muito útil para downloads de arquivos grandes, para programas **P2P** que constroem o arquivo através de pedaços e estão sujeitos a corrupção dos mesmos. Como autenticação de login é utilizada em vários sistemas operacionais unix e em muitos sites com autentificação.

CRIPTOGRAFIA

- SHA (Secure Hash Algorithm) está relacionada com as funções criptográficas. A função mais usada nesta família, a SHA-1, é usada numa grande variedade de aplicações e protocolos de segurança, incluindo TLS, SSL, PGP, SSH, S/MIME e IPSec. SHA-1 foi considerado o sucessor do MD5. Ambos têm vulnerabilidades comprovadas. Em algumas correntes, é sugerido que o SHA-256 ou superior seja usado para tecnologia crítica. Os algoritmos SHA foram desenhados pela National Security Agency (NSA) e publicados como um padrão do governo Norte-Americano.

- O primeiro membro da família, publicado em 1993, foi oficialmente chamado SHA; no entanto, é frequentemente chamado SHA-0 para evitar confusões com os seus sucessores. Dois anos mais tarde, SHA-1, o primeiro sucessor do SHA, foi publicado. Desde então quatro variantes foram lançadas com capacidades de saída aumentadas e um design ligeiramente diferente: SHA-224, SHA-256, SHA-384, e SHA-512 — por vezes chamadas de SHA-2.

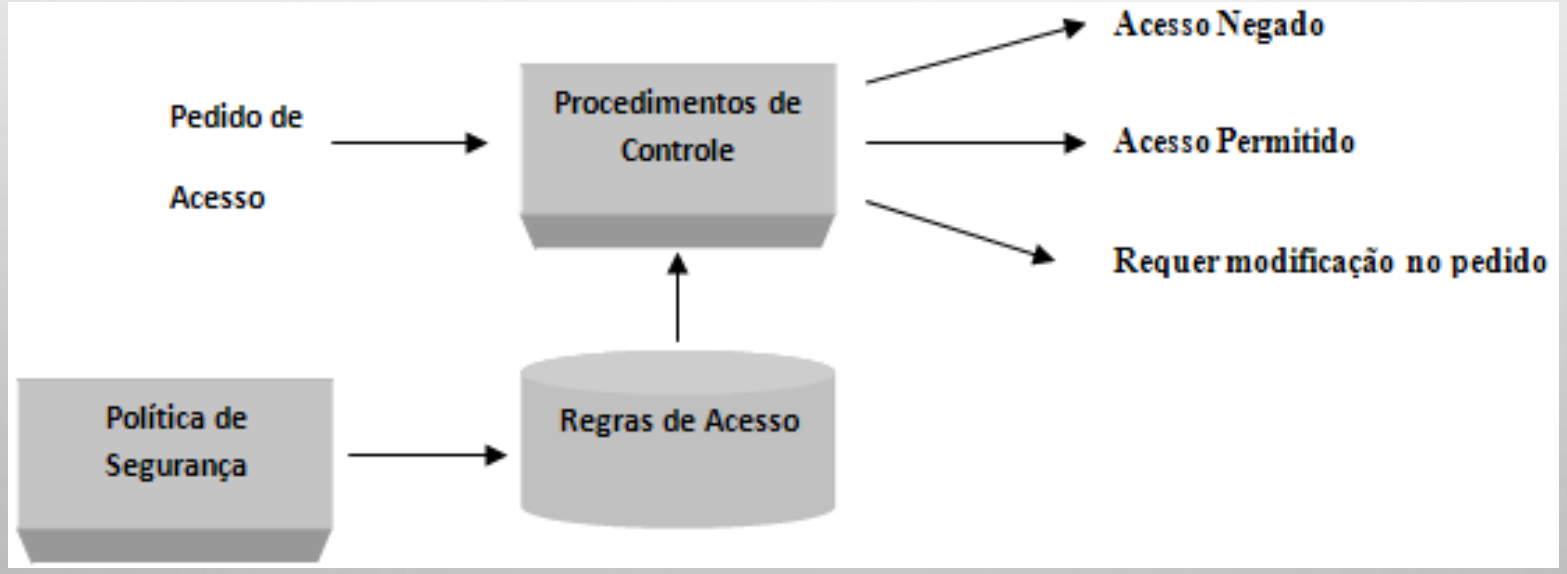
SEGURANÇA DE ACESSO (USUÁRIOS E APLICAÇÕES)

- A preocupação com a criação e manutenção de ambientes seguros têm sido tarefas cruciais de administradores de redes, de sistemas operacionais e de bancos de dados.
- Pesquisas mostram que boa parte dos ataques, roubos de informações e acessos não-autorizados são feitos por pessoas que pertencem à organização alvo.
- Isso faz com que estes profissionais se desdobrem para criar e usar artifícios para eliminar os acessos não-autorizados ou minimizar as chances de sucesso das tentativas de invasão (internas ou externas).
- Os controles de acesso em sistemas de informação devem assegurar que todos os acessos diretos ao sistema ocorram exclusivamente de acordo com as modalidades e as regras pré-estabelecidas, e observadas por políticas de proteção.

SEGURANÇA DE ACESSO (USUÁRIOS E APLICAÇÕES)

A

figura abaixo apresenta um sistema de controle de acesso incluindo assuntos (usuários e processos) que alcançam objetos (dados e programas) com as operações (ler, escrever e executar).



SEGURANÇA DE ACESSO (USUÁRIOS E APLICAÇÕES)

A figura mostra um sistema de controle de acesso composto basicamente de dois componentes:

- as **Políticas de Acesso,** que indica as modalidades e tipos de acesso a serem seguidas, e
- os **Procedimentos de Acesso**, que, com base nas regras de acesso, os pedidos de acesso podem ser permitidos, negados ou podem ser pedidas modificações no pedido de acesso.

Grande parte dos SGBDs atuais controla o acesso aos dados armazenados através de **Listas de Controle de Acesso** (*Access Control List,* ACL) .

As ACLs são tabelas especiais que possuem informações sobre os **privilégios** que cada usuário pode ter em determinado banco de dados.

SEGURANÇA DE ACESSO (USUÁRIOS E APLICAÇÕES)

- Quando um usuário se conecta ao banco de dados “sua identidade” é determinada pela máquina de onde ele conectou e o nome de usuário que ele especificou.
- O sistema concede **privilégios** de acordo com sua identidade e com o que ele deseja fazer.
- Assim é possível que usuários provenientes de diferentes lugares da internet possuam o mesmo nome de usuário e privilégios totalmente diferentes um do outro.
- O controle de acesso é feito em duas etapas: primeiramente o servidor confere se você pode ter acesso ou não, em seguida, assumindo que você pode conectar, o servidor verifica cada requisição feita para saber se você tem ou não privilégios suficientes para realizar a operação. Por exemplo, se você tentar selecionar linha de uma tabela em um banco de dados ou apagar uma tabela do banco de dados, o servidor se certifica que você tem o privilégio **select** para a tabela ou o privilégio **drop** para o banco de dados.

SQL INJECTION

O que é ?

A Injeção de SQL, mais conhecida através do termo americano SQL Injection, é um tipo de ameaça de segurança que se aproveita de falhas em sistemas que interagem com bases de dados via SQL.

A injeção de SQL ocorre quando o atacante consegue inserir uma série de instruções SQL dentro de uma consulta (query) através da manipulação das entrada de dados de uma aplicação.

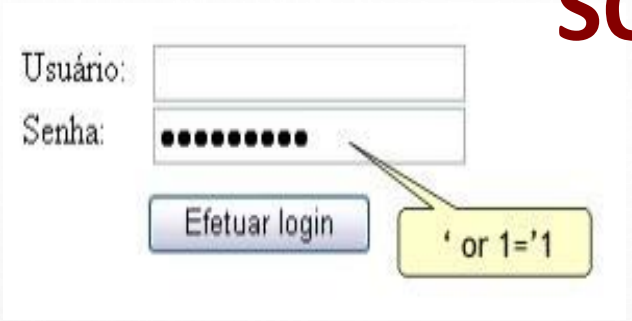
SQL INJECTION

- Para ilustrar o conceito de SQL Injection, a seguinte simulação pode ser realizada. Imaginemos que um script de validação de acesso de usuários tenha sido desenvolvido como segue:

```
1  <?php
2
3    $usuario = $_POST['usuario'];
4    $senha = $_POST['senha'];
5
6    $query_string ="SELECT * FROM usuarios
7                    WHERE codigo = '{$usuario}' AND senha = '{$senha}'";
8
9  ?>
10
```

- Nas linhas 3 e 4, as variáveis $usuario e $senha, respectivamente, recebem o conteúdo submetido por um formulário através do método POST. Eis a fonte do problema.
- Suponha que a seguinte entrada tenha sido informada no campo usuário no formulário chamador do script de validação.

SQL INJECTION



- Logo, a query string resultante será:

- Se nenhuma outra validação for realizada, o usuário mal intencionado terá efetuado login no sistema, sem ao menos informar um usuário contido na tabela. Isto foi possível pois o valor de entrada informado não recebeu o tratamento devido, sendo adicionado à instrução para ser executado. Vale ressaltar que as validações apresentadas no exemplo são apenas ilustrativas, havendo a necessidade de checagens mais eficazes para um script de validação de acesso.

SQL INJECTION

Impossibilitando o uso de SQL Injection

- Para que se esteja livre da utilização da SQL Injection, certas providências devem ser tomadas. Algumas das ações serão realizadas no servidor de banco de dados, outras devem ser garantidas pelo código fonte.
- Deve-se tomar cuidado com a configuração do usuário que estabelece a conexão com o banco de dados.
- O ideal é que as permissões de acesso deste usuário estejam restritamente limitadas às funções que irá realizar, ou seja, para a exibição de um relatório, a conexão com o banco de dados deve ser realizada por um usuário com permissões de leitura e acesso somente às tabelas necessárias para sua operação.

SQL INJECTION

- Todos os valores originados da coleta de dados externos, devem ser validadas e tratadas a fim de impedir a execução de eventuais instruções destrutivas ou operações que não sejam as esperadas.
- Um tratamento básico para a execução de querys com variáveis contendo valores informados pelo usuário:

```php
<?php

    $usuario = $_POST['usuario'];
    $senha = $_POST['senha'];
    $usuario_escape = addslashes($usuario);
    $senha_escape = addslashes($senha);

    $query_string =  "SELECT * FROM usuarios
                      WHERE codigo = '{$usuario_escape}'
                      AND senha = '{$senha_escape}'";

?>
```

SQL INJECTION

- Com a utilização da função addslashes() será adicionada uma barra invertida antes de cada aspa simples e aspa dupla encontrada, processo conhecido como escape. Se a diretiva de configuração do PHP magic_quotes_gpc estiver ativada, o escape é realizado automaticamente sobre os dados de COOKIES e dados recebidos através dos métodos GET e POST. Neste caso, não deve ser efetuado o tratamento com addslashes(). A função get_magic_quotes_gpc(), disponível nas versões do PHP a partir da 3.0.6, retorna a configuração atual da diretiva magic_quotes_gpc.
- Abaixo, a query string resultante da aplicação do tratamento mencionado:

SQL INJECTION

- Em muitos bancos de dados, existem funções específicas para o tratamento de variáveis em query strings, o que diminui a compatibilidade do código fonte para operação com outros sistemas de banco de dados.
- Outra dica importante é evitar a exibição das mensagem de erro em um servidor de aplicação em produção, pois geralmente nos erros ou alertas são exibidos caminhos de diretórios do sistema de arquivos e informações à respeito do esquema do banco de dados, podendo comprometer a segurança do sistema.

SEGURANÇA CONTRA FALHAS

(RECOVERY)

- A recuperação/tolerância a falhas tem por objetivo restaurar o BD para um estado de integridade, após a ocorrência de uma falha;

- Os mecanismos de recuperação baseiam-se na utilização de formas de redundância que quase duplicam o BD, utilizando: *Backup* e *transaction log*

MANUTENÇÃO DE HISTÓRICO DE ATUALIZAÇÕES(LOG) E BACKUPS DO BD

- Os *backups* são cópias de segurança do BD, que são executados periodicamente e constituem um ponto de partida para a recuperação do BD após a ocorrência de uma falha, independentemente da sua gravidade;

- Refletem uma situação passada, donde a reposição a partir de um *backup* implica perder todas as atualizações posteriores à realização do mesmo.

MANUTENÇÃO DE HISTÓRICO DE ATUALIZAÇÕES(LOG) E BACKUPS DO BD

- Uma forma de minimizar esta situação é aumentando a periodicidade dos *backups*, o que é um processo pesado, consumidor de recursos e que pode obrigar a paradas dos sistemas;

- A periodicidade dos *backups* depende do valor dos dados, do seu volume e da frequência com que são acedidos e alterados;

MANUTENÇÃO DE HISTÓRICO DE ATUALIZAÇÕES(LOG) E BACKUPS DO BD

- O *backup* é um mecanismo de reposição do BD.
- Os *transaction logs* são mecanismos de repetição das transacções ocorridas desde o último *backup (rollforward);*
- Normalmente para se refazer uma transação é necessário o ficheiro de *transaction log*, onde está guardada uma identificação da transação e uma cópia dos dados atualizados por ela (*after image*);

MANUTENÇÃO DE HISTÓRICO DE ATUALIZAÇÕES(LOG) E BACKUPS DO BD

- Sendo esta a forma mais comum de resolver os problemas provocados por uma falha, têm que existir outros mecanismos que permitam o *roll back* das transacções não terminadas, ocorridos durante a execução não sucedida das mesmas;
- Donde o ficheiro *transaction log* necessita igualmente de guardar os dados anteriores ao início da execução da transacção (*before-images)*

MANUTENÇÃO DE HISTÓRICO DE ATUALIZAÇÕES(LOG) E BACKUPS DO BD

É da conjugação destes dois mecanismos - *backups* e *transaction logs* , que se garantem duas das características fundamentais das transações:

- **Atomicidade** (desfazendo uma transação não sucedida)
- **Persistência** (refazendo os efeitos de uma transação bem sucedida)

TIPO DE FALHAS

Falha de disco - o(s) disco(s) onde o BD está armazenado fica(m) inutilizado(s). É a falha considerada mais grave e que obriga à reconstrução de todo o SGBD;

Falha de sistema - pode resultar de problemas de hardware ou software, não sendo possível garantir a validade dos dados. Implica repor a BD a partir do seu último estado de integridade.

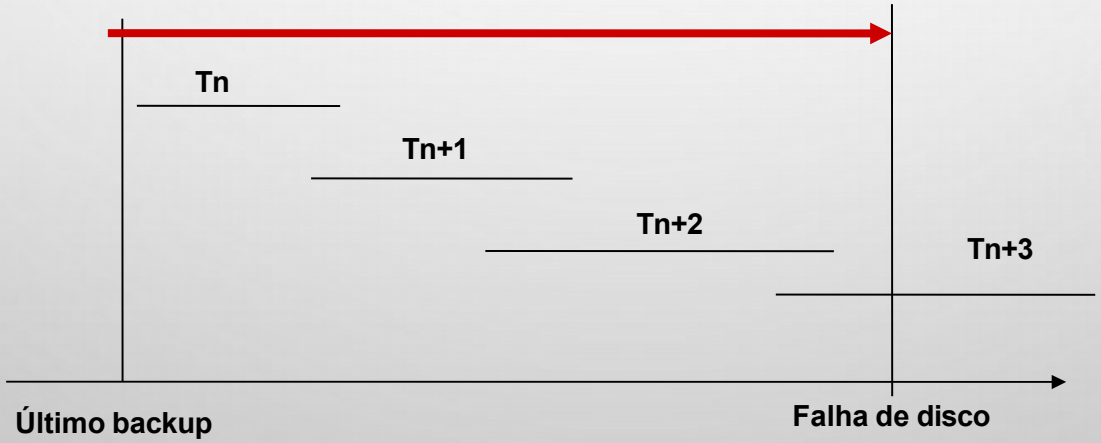
TIPO DE FALHAS

Falha de transação - é a mais inofensiva e recupera-se recorrendo ao ficheiro *transaction log* e às *before images* da transação que não foi bem sucedida.

Em qualquer processo de recuperação recorre-se ao *rollback* das transações efetuadas até ao momento em que os *transaction log* e os ficherios da base estão sincronizados, para se poderem desfazer todas as transações decorridas desde então.

TIPO DE FALHAS

Esse momento coincide com o último *backup*, o qual se muito afastado no tempo, obriga a um processo moroso e complexo de recuperação do BD.



TIPO DE FALHAS

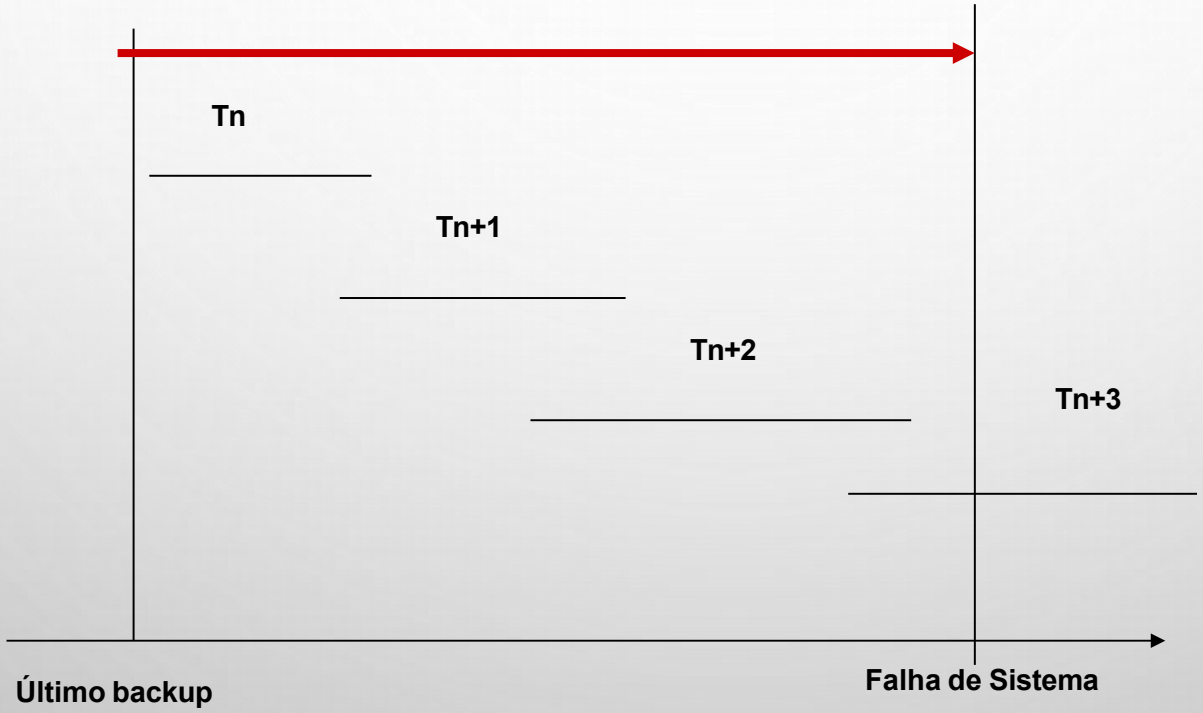
Para evitar este tipo de situações, recorre-se a marcas de segurança, conhecidas por ***checkpoints***.

Basicamente, para reduzir o número de acessos aos discos, nos ficheiros de *transaction log* as atualizações são realizadas na memória RAM, em *buffers*, sendo posteriormente escritos em disco;

TIPO DE FALHAS

- QUANDO OCORRE UMA FALHA, O CONTEÚDO DESSES **BUFFERS** PODE PERDER-SE, OU PELO MENOS PODE NÃO EXISTIR UMA GARANTIA DE VALIDADE DO SEU CONTEÚDO;
- ASSIM OS **CHECKPOINTS** REGISTRAM OS MOMENTOS EM QUE O CONTEÚDO DOS **BUFFERS** FOI ESCRITO NOS DISCOS, DEFININDO-SE UM MOMENTO EM QUE O TRANSACTION LOG E O BD ESTÃO

TIPO DE FALHAS



AUDITORIA EM BANCO DE DADOS

- É o conjunto de ações para verificar o que os usuários estão fazendo.
- Muitas empresas fazem isso para os fins de segurança. Isso é para garantir que usuários não autorizados não estão acessando uma parte do banco de dados ou a estrutura principal que não é permitida.
- A maioria das informações críticas de uma empresa, 90% ou mais, é mantida no banco de dados.
- É por isso que a auditoria do banco de dados é tão crucial para a proteção da segurança.
- Se determinada informação é comprometida, ela pode ser crítica para suas operações.

SEGURANÇA EM BANCO DE DADOS LIVRES (MARIADB)

- O MariaDB 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 MariaDB já dá suporte a transações, derrubando este impecílio à sua utilização. Mais tarde, com a introdução das tabelas InnoDB, o MariaDB ganhou mais um poderoso recurso: integridade referencial.
- Assim o MariaDB 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.

SEGURANÇA NO SISTEMA (MARIADB)

- 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 MariaDB.
- Apesar de implementar um sistema de validação robusto, o MariaDB não tem como controlar acessos que deveriam ser bloqueados pelo sistema operacional.
- Acesso a arquivos, permissóes de usuários do sistema, 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 MariaDB deve ter acesso à aos arquivos de dados do MariaDB.

SISTEMA DE ARQUIVOS (MARIADB)

- 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 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 MariaDB 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.

SISTEMA DE ARQUIVOS (MARIADB)

Já a permissão de leitura merece uma explicação mais detalhada.

- Ao bloquear o acesso à escrita nos arquivos do MariaDB, 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 MariaDB 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.

USUÁRIOS DO SISTEMA (MARIADB)

- Para acessar o banco de dados, não é necessário criar uma conta no sistema operacional, pois o MariaDB 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 MariaDB, 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?

USUÁRIOS DO SISTEMA (MARIADB)

- Existem algumas situações que o admistrador deve dar acesso ao sistema de arquivos para o dono do banco de dados, pois algumas funções no MariaDB precisam deste acesso (LOAD DATA por exemplo).
- Se este for o caso, o administrador deve ter o cuidado para não deixar aberto acesso a arquivos importantes. É óbvio, mas existem administradores 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.

USUÁRIOS DO SISTEMA (MARIADB)

- Um administrador 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 MariaDB 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 MariaDB antes mesmo de fazer o servidor MariaDB funcionar.

SISTEMA DE AUTENTICAÇÃO (MARIADB)

- O MariaDB 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 MariaDB.

- Pela natureza dos dados que este banco de dados armazena, ele deve ter o acesso permitido apenas para o usuário root do MariaDB.

- Os usuários comuns não necessitam de acessar este banco de dados, principalmente a tabela user, onde estão armazenadas as senhas dos usuários.

SISTEMA DE AUTENTICAÇÃO (MARIADB)

- Para aceitar a conexão de um usuário, o MariaDB 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 MariaDB.
- A melhor escolha é usar os comandos **GRANT/REVOKE**, pois o MariaDB 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 manualmente além do risco de manipular dados de forma errada, você pode se esquecer de executar o comando **FLUSH PRIVILEGES** para tornar as alterações ativas.

SISTEMA DE AUTENTICAÇÃO (MARIADB)

- 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 administrador 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.

SISTEMA DE AUTENTICAÇÃO (MARIADB)

- 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 MariaDB 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.

SISTEMA DE AUTENTICAÇÃO (MARIADB)

- Você poderia ainda usar o curinga ’%’ indicando que o usuário pode se conectar 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 administrador entenda pelo menos basicamente o funcionamento do sistema de privilégios do MariaDB 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 MariaDB para ver os privilégios disponíveis e como utilizá-los de forma correta.

CONEXÃO VIA REDE (MARIADB)

Ao conectar ao servidor MariaDB localmente, tendo um sistema bem configurado o MariaDB 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 MariaDB 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.

CONEXÃO VIA REDE (MARIADB)

- 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 MariaDB 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 MariaDB 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 MariaDB 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 MariaDB 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 MariaDB 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 MariaDB dá todas as informações passo a passo para criar certificados para o MariaDB e como configurar o MariaDB para utilizar acesso seguro via SSL.

Se o servidor MariaDB deve ser visto apenas na rede local, o acesso externo deve ser bloqueado. Você pode fazer isto com o próprio esquema de privilégios do MariaDB, 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 MariaDB o próprio firewall se encarrega de filtrar o acesso. Dessa forma a porta de acesso ao MariaDB será fechada para conexões externas.

CONSEQUENCIAS

- Sabe o que essas empresas tem em comum ?

CONSEQUENCIAS DE BANCO DE DADOS NÃO SEGUROS.

INVASÃO EM BANCO DE DADOS COLOCA EM RISCO 2 MILHÕES DE CLIENTES DA HONDA

Clientes da Honda estão correndo risco. Não, não se trata de um recall nos automóveis fabricados, mas sim uma invasão no banco de dados de e-mails dos clientes da empresa. Cerca de dois milhões de endereços foram roubados, além de informações pessoais como endereço, senha e modelo e identificação do veículo.

A empresa está investigando como o crime aconteceu e tentando identificar os principais culpados. Até agora, quem está na mira da investigação é a empresa de marketing Silverpop Systems, responsáveis pelas newsletters da fabricante.

A Honda avisou todos os clientes em um comunicado oficial enviado ao e-mail de cada um. Afinal, com posse de todos esses dados, é simples receber um comunicado como se fosse o representante Honda local, com todos os seus dados e pedindo mais informações pessoais. Um perigo.

Fonte: http://bit.ly/gJoUcW

CONSEQUENCIAS DE BANCO DE DADOS NÃO SEGUROS.

Utilizando técnica de injeção de SQL, dupla de invasores obteve lista de credenciais de acesso de diversos domínios ligados ao produto da Oracle.

O site MySQL.com, para clientes do banco de dados adquirido pela Oracle com a compra da Sun, foi aparentemente comprometido no fim de semana por uma dupla de hackers que publicaram nomes de usuários - e, em alguns casos, senhas - dos usuários dos site.

Identificando-se como "TinKode" e "Ne0h", os hackers afirmaram ter utilizado - ironicamente - um ataque de injeção SQL, mas não forneceram detalhes sobre a operação. Os domínios vulneráveis foram listados como www.mysql.com, www.mysql.fr, www.mysql.de, www.mysql.it e www-jp.mysql.com.

De acordo com um post da lista de discussão de bugs Full Disclosure, o MySQL.com mantém diversos bancos de dados internos em um servidor web Apache. A informação postada incluiu diversos códigos hash de senhas - algumas das quais já foram quebradas.

CONSEQUENCIAS DE BANCO DE DADOS NÃO SEGUROS.

Entre as credenciais que constavam de uma lista de dados publicada no Pastebin estavam senhas para diversos usuários dos bancos de dados MySQL instalados no servidor, e as senhas admin para blogs corporativos de dois ex-funcionários da MySQL : o ex-diretor de gerenciamento de produtos Robin Schumacher e o ex-vice-presidente de relações com a comunidade, Kaj Arnö.

Schumacher é atualmente diretor de estratégia de produtos na EnterpriseDB, enquanto Arnö trabalha como vice-presidente executivo de produtos na SkySQL. O blog de Schumacher não foi tocado desde junho de 2009; o de Arnö está inativo desde janeiro de 2010.

A Oracle, que obteve o controle da MySQL com a aquisição da Sun Microsystems em abril de 2009, não comentou o incidente.

Uma empresa de segurança que monitora ataques feitos a sites, Sucuri, aconselhou aos usuários com uma conta no MySQL.com a mudar suas senhas o mais rapidamente possível, especialmente se eles usam a mesma senha em vários sites.

Fonte: http://bit.ly/eU3CN6

RECOMENDAÇÕES PARA UM AMBIENTE SEGURO

- Nunca fornecer a ninguém (exceto aos usuários administrativos) acesso a tabela da ACL, caso uma pessoa tenha acesso as senhas de acesso da ACL, ela poderá facilmente se conectar ao banco com qualquer conta;
- Conceder apenas os privilégios necessários para cada usuário, nunca mais do que isso;
- Não manter senhas em texto puro no banco de dados, em vez disso, utilizar alguma função de criptografia de via única, com SHA1 ou MD5;
- Não escolher senhas que contenham palavras existentes em dicionários;
- Utilizar um *firewall*;
- Não confiar em nenhum dado inserido pelos usuários, muitos deles podem tentar atacar o sistema inserindo caracteres especiais nas entradas dos formulários contendo algum.

REFERÊNCIAS

KORTH, Henry F.; SILBERSCHATZ, Abraham. Sistemas de Banco de Dados. 3 ed.São Paulo. Makron Books, 1999.

MANUAL DE REFERÊNCIA DO MariaDB. Como o Sistema de Privilégios Funciona. 2008b Disponível em < http://dev.MariaDB.com/doc/refman/4.1/pt/privileges.html>. Acesso em 01 abr. 2011,

Marcos SÊMOLA . A importância da gestão de segurança da informação. 2003. Disponível em <http://www.linorg.cirp.usp.br/SSI/SSI2003/Palest/P03-Apresentacao.pdf>. Acesso em 01 abr. 2011.

SQL Injection FAQ. Disponível em <http://www.sqlsecurity.com/FAQs/SQLInjectionFAQ/tabid/56/Default.aspx>. Acesso em 01 abr. 2011.

Maico KRAUSE. Segurança em Banco de Dados. Disponível em <http://bit.ly/i9diim>. Acesso em 20 mar. 2011.

Wikipédia. Segurança da informação. Disponivel em <http://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_informa%C3%A7%C3%A3o>. Acesso em 25 mar. 2011.

SQL Injection. Disponivel em <http://imasters.com.br/artigo/5179/sql_injection_no_php_o_que_e_e_como_se_proteger>. Acesso em 01 abr. 2011.

*** Alguns dos links estão usando encurtador de URLs***