Proteção de dados em MySQL e MariaDB

GTS-36, 2021

Danton Nunes <danton.nunes@inexo.com.br>

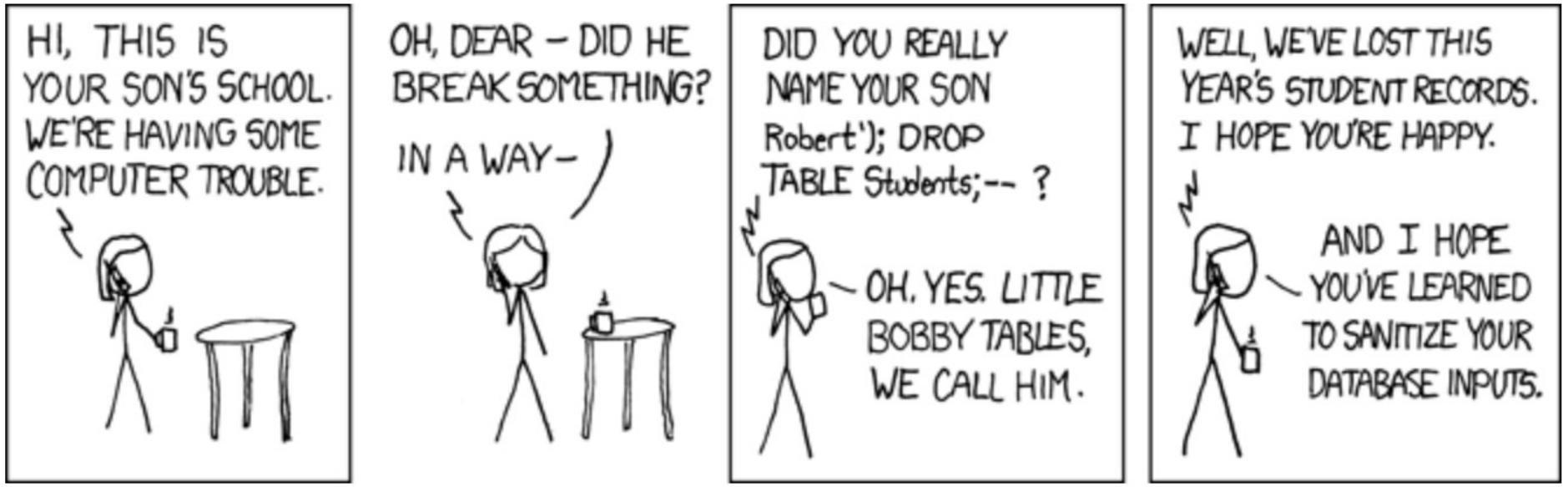
Resumo

- A LGPD trouxe novos e complexos requisitos de segurança de dados.
- A maioria das aplicações na Web usa um modelo intrinsecamente inseguro.
- Há o risco de vazamento de dados protegidos por causa disso.
- O modelo proposto visa sanar esse problema

O modelo convencional

- O cliente acede à página Web, identificado ou não.
- O servidor web aciona um script (geralmente em PHP) que consulta/modifica o banco de dados.
- Esse script faz a conexão com o banco com um usuário/senha único para todas as conexões.
- Nesse usuário único é que mora o perigo, pois ele tem acesso à tabelas completas, não só às linhas que interessam ao usuário.

O perigo que se corre



https://xkcd.com/327/

- **Não temos evidências desse tipo de vulnerabilidade em nosso sistema**
- **A ausência de evidência não é evidência da ausência (Carl Sagan)**

O modelo convencional

- Um ataque bem sucedido de injeção de SQL pode resultar no download de um cadastro inteiro de dados pessoais.
- Você não pode determinar quem fez isso porque é um só usuário do ponto de vista do banco de dados.
- Resumindo, um mato sem cachorro!

usuário web = usuário SQL

- Para aplicações em que o usuário se identifica (login), cada usuário deve ser também usuário do banco de dados.
- No caso do *MySQL* e *MariaDB* a tabela `mysql.users` é o lugar certo para se cadastrar usuários!

```
MariaDB [mysql]> create user `493`;
Query OK, 0 rows affected (0.015 sec)
```

usuário web = usuário SQL

- E também para guardar senhas!

```
MariaDB [mysql]> set password for `493`=password('senha');
Query OK, 0 rows affected (0.018 sec)
```

  e as senhas ficam cifradas!

- Também é o lugar certo para dar os privilégios adequados

```
MariaDB [mysql]> grant select on dados.cliente,dados.faturas,dados.suporte to `493`;
Query OK, 0 rows affected (0.012 sec)
```

  e a autenticação é mais fácil: conectou no banco = OK!

Vistas limitadas das tabelas

- Em vez do cliente ver toda a tabela de cadastro, pode-se definir uma vista (view) restrita às linhas que lhe dizem respeito

```
create sql security definer view cliente1 as
    select * from cliente where user() like concat(id,'@%')
    with cascaded CHECK option;
```

user() retorna o usuário corrente, de modo que cliente1 só tem as linhas em que id = user@...

Vistas limitadas das tabelas

- Se a senha do usuário for comprometida e usada em um ataque bem sucedido de injeção de SQL, o atacante vai capturar somente uma linha do cadastro, e é a que ele já conhece, em vez da tabela toda!
- E no log fica registrado quem fez o que e quando!

Logs: registrar **TUDO**

- Normalmente o log de um gerenciador de BD registra as operações que alteram dados, com a LGPD em vigor pode ser interessante registrar também os selects!
- Note que o log também contém dados protegidos, portanto deve ser tratado adequadamente.

E as aplicações?

- Nas aplicações, o login fica mais fácil. Pegue as credenciais que o usuário passou e tente conectar no banco de dados com elas. Se passou, elas estão corretas. Duas pauladas no mesmo coelho (ou seria ao contrário?)
- No resto da aplicação não precisa mexer em nada, se as vistas forem bem feitas.

Conclusão

- Usuário da aplicação = usuário do banco de dados:
  - facilita a autenticação, senhas cifradas,
  - identifica autoria das operações no log.
- Vistas restritas das tabelas sensíveis:
  - protege o sistema de vazamentos por ataques de injeção de SQL.

Referência e Perguntas

- Este trabalho nasceu deste paper:
  http://zaphod.inexo.com.br/~danton/paper-data-protection.html

?