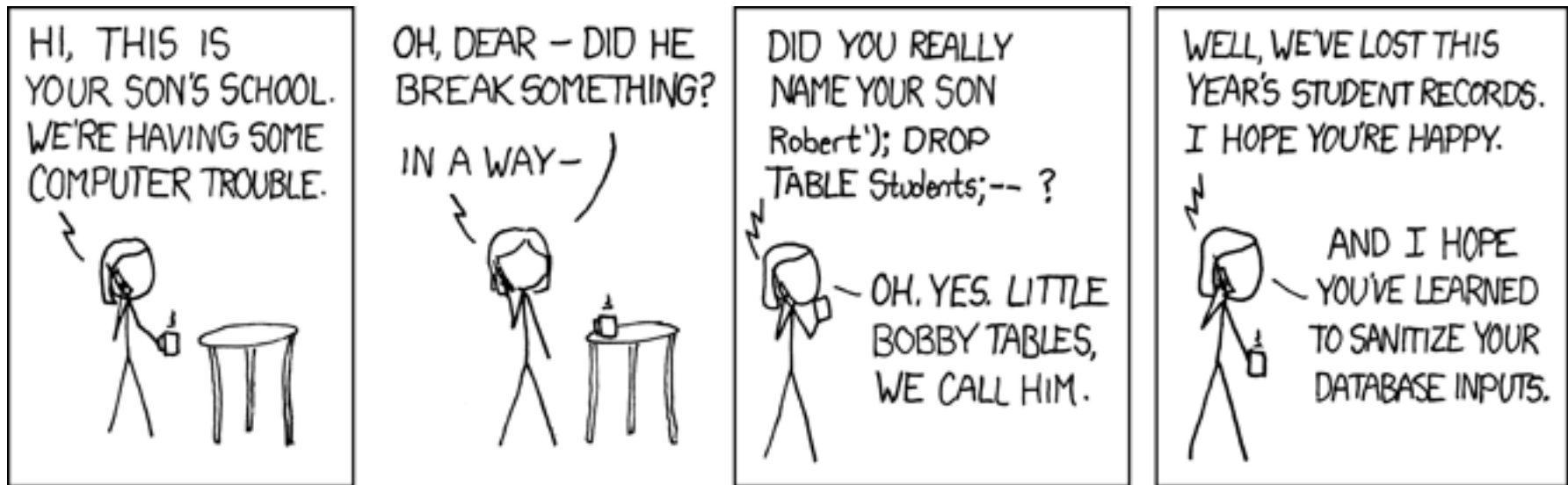


Avoiding SQL Injection

Alex Beamish / January 2010
Toronto Perlmongers

Visual aid / break the ice



The problem

```
my $q = CGI->new;
my $p = $q->param("password");
my $query="SELECT user_id FROM users WHERE password=$p";
my $sth = $dbh->prepare($query) or die "Error preparing $query: " . $dbh->errstr;
$sth->execute or die "Error getting user: " . $sth->errstr;
my $user = $sth->fetchrow_hashref;
...
```

Problem: The password parameter contains data that compromises the query.

Solutions: As usual, there's more than one way to do it.

Placeholders

```
my $q = CGI->new;  
my $p = $q->param("password");  
my $query="SELECT user_id FROM users WHERE password=?";  
my $sth = $dbh->prepare($query) or die "Error preparing $query: " . $dbh->errstr;  
$sth->execute($p) or die "Error getting user: " . $sth->errstr;  
my $user = $sth->fetchrow_hashref;
```

User privileges

[] Have the web application access the database using an account with limited privileges, that is, one that can SELECT, INSERT and UPDATE, but not one that can do DROP TABLE or DROP DATABASE operations.

Database configuration

[] Have the tables configured as read-only. For example, transaction tables that are split every few months can be made read-only.

Use SQL::Abstract

```
my $fields = '(' . join( ',', ( keys %{$args}, keys %fields ) ) . ')';  
my $fieldCount = ( scalar keys %{$args} ) + ( scalar keys %fields );  
my $valueQs = '(' . join( ',', ('?') x $fieldCount ) . ')';  
my $cmd = "INSERT INTO table $fields VALUES $valueQs";
```

```
my $sth = $dbh->prepare($cmd)  
    or croak "Unable to prepare command $cmd: " . $dbh->errstr;  
$sth->execute( values %{$args}, values %dates )  
    or croak "Unable to execute command $cmd: " . $sth->errstr;  
$sth->finish;
```

becomes ..

Use SQL::Abstract / 2

```
my $sql = SQL::Abstract->new( array_datatypes => 1 );  
my ( $cmd, @bindVars ) = $sql->insert( 'table', \%data );  
my $rows = $dbh->do( $cmd, {}, @bindVars )  
    or croak "Unable to do command $cmd: " . $dbh->errstr;
```

Conclusions

- [] SQL with variable interpolation: BAD!
- [] SQL using placeholders: Good!
- [] Using SQL::`Abstract`: Not always necessary, but can be .. Divine!