Malicious Page:

\[ \text{img src} = "http://facebook.com/change-rel-status?complicit} \]

Submit a form:

- **GET request**
  - GET /page? key=val & key=...
  - Headers...

- **POST request**
  - POST /page
  - Headers...
  - key=val & key=...

Countermeasures:

- Use POST instead of GET
  - Idea: \( \text{img src} = \ldots \) will always make browser do a GET request
  - Good practice, but doesn't fix entire problem

- Can trigger POST requests from a page (using Javascript)

- **Referral Header**: browser reports page that user clicked to get here
  - Idea: check Referer header before performing action
  - Problem: this header is optional, and there are legit reasons why it might not be there
Double-submit cookie [Zeller-Felten]

Idea: every form submission should contain a "secret" (hard to guess) that only the legit user should know or only appears when user submits legitimate form.

```
legit user
    login
        Session/auth cookie
        "CSRF token" cookie
            Set-Cookie: csrf = <random value>

    request form
        CSRF token
            form
                <form ...
                    <input type=hidden name=csrf
                        value = <random value>
                ...

    submit form
        CSRF token cookie
            POST data csrf: <random value>
        check if =
```
**SQL Injection!!!**

SQL = language used to communicate w/ database (relational) database = collection of tables

table =

**Ex:** employees table

<table>
<thead>
<tr>
<th>name</th>
<th>ssn</th>
<th>dept</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>123</td>
<td>ECEs</td>
<td>12</td>
</tr>
</tbody>
</table>

rows / columns (fields)

columns have "types"

<SQLite demo>

CREATE TABLE ... 
INSERT INTO ... 
SELECT

```
SELECT * FROM table_name
```

"all columns"

```
SELECT age, name, age + 1 ...
```

```
SELECT .... WHERE age > 30
```