Web Programming Step by Step

Lecture 26
Web Security

1. The "security mindset"

- **security mindset**
- some basic web attacks
- breaking and securing an example page

CSE <= 190M

- until now, we have assumed:
  - valid user input
  - non-malicious users
  - nothing will ever go wrong
- this is unrealistic!

The real world

- in order to write secure code, we must assume:
  - invalid input
  - evil users
  - everybody is out to get you
- trust nothing

2. Some basic web attacks

- security mindset
- **some basic web attacks**
- breaking and securing an example page

HTML injection

a flaw where a user is able to inject arbitrary HTML content into your page

- why is this bad? it allows others to:
  - disrupt the flow/layout of your site
  - put words into your mouth
  - (possibly) run JavaScript on your users' computers
- kinds of injected content:
  - annoying: `results.php?name=<blink>lololol</blink>`
  - malicious and harmful: `onlinebanking.php?text=<script>transferMoneyTo("Evil Kevin", 1000, "USD");</script>`
    - injecting JavaScript content is called **cross-site scripting** or XSS
- example: magic 8-ball
  - https://webster.cs.washington.edu/kwal/lecture26/8ball/

Securing against HTML injection

- one idea: disallow harmful characters
  - HTML injection is impossible without < >
  - can strip those characters from incoming input
  - or, just reject the entire request if they are present
- better idea: allow them, but **escape** them
  - < > → < >
  - PHP's `htmlspecialchars` function escapes HTML characters:

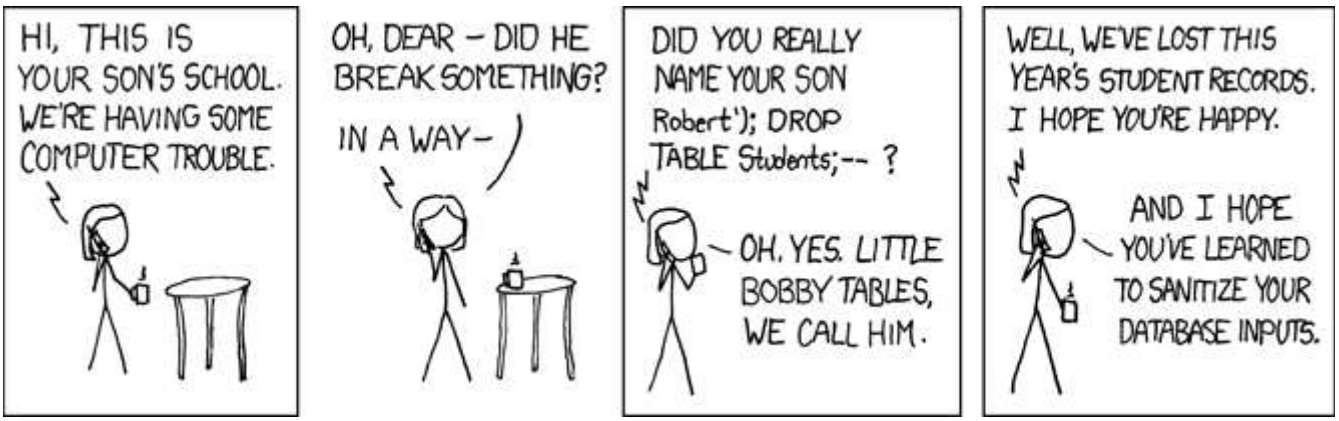
```php
<?= htmlspecialchars($username) ?>
```

SQL injection

a flaw where the user is able to inject arbitrary SQL commands into your query

- `$query = "SELECT name, ssn, dob FROM users WHERE username = '$username' AND password = '$password'";`
  - Password: `' OR '1'='1`
- `$query = "SELECT name, ssn, dob FROM users WHERE username = '$username' AND password = '' OR '1'='1'";`
  - What will the above query return? Why is this bad?
- example: simpsons grade lookup
  - https://webster.cs.washington.edu/kwal/lecture26/grades/

Securing against SQL injection



- similar to securing against HTML injection, escape the string before you include it in your query
- use the PHP `mysql_real_escape_string` function

```
$username = mysql_real_escape_string($_REQUEST["username"]);
$password = mysql_real_escape_string($_REQUEST["password"]);
$query = "SELECT name, ssn, dob FROM users
WHERE username = '$username' AND password = '$password'";
```
PHP

3. Breaking and securing an example page

- PHP/SQL review
- some basic web attacks
- **breaking and securing an example page**