Web Security Basics

CSE 190 M (Web Programming) Spring 2008
University of Washington

Lecture outline

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

PHP/SQL review

let's write an unsecure page using PHP and SQL

Recall: PHP MySQL functions

- `mysql_connect("`*server*`", "`*username*`", "`*password*`")`
  connects to the given server; returns FALSE on failure
- `mysql_select_db("`*database*`")`
  chooses the given database; returns FALSE if not found
- `mysql_query("`*query*`")`
  executes the given SQL query on the currently selected database; returns a result-set object, or FALSE if query fails
- `mysql_fetch_array(`*results*`)`
  returns one row from the given query result set as an associative array, or FALSE when no more rows remain
- `mysql_error()`
  returns a string representing the most recent MySQL-related error that has occurred

Complete PHP MySQL example

```
# connect to world database on local computer
$db = mysql_connect("localhost", "traveler", "packmybags");

mysql_select_db("world");

# execute a SQL query on the database
$results = mysql_query("SELECT * FROM Countries WHERE population > 100000000


# loop through each country
while ($row = mysql_fetch_array($results)) {
?>
  <li><?= $row["name"] ?>, ruled by <?= $row["head_of_state"] ?></li>
<?php
}
?>
```

PHP

Complete example w/ error checking

```php
# connect to world database on local computer
$db = mysql_connect("localhost", "traveler", "packmybags");
check_result($db);
check_result(mysql_select_db("world"));

# execute a SQL query on the database
$results = mysql_query("SELECT * FROM Countries WHERE population > 100000000
check_result($results);

# loop through each country
while ($row = mysql_fetch_array($results)) {
?>
  <li><?= $row["name"] ?>, ruled by <?= $row["head_of_state"] ?></li>
<?php
}

# stops the page if any MySQL error occurred
function check_result($value) {
  if (!$value) {
    die("SQL error occurred: " . mysql_error());
  }
}
?>
```

PHP

Simpsons database w/ passwords

students

id	name	email	password
123	Bart	bart@fox.com	bartman
404	Ralph	ralph@fox.com	catfood
456	Milhouse	milhouse@fox.com	fallout
888	Lisa	lisa@fox.com	vegan

courses

id	name	teacher_id
10001	Computer Science 142	1234
10002	Computer Science 143	5678
10003	Computer Science 190M	9012
10004	Informatics 100	1234

grades

student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

teachers

id	name
1234	Krabappel
5678	Hoover
9012	Stepp

Web Security

breaking and securing web pages

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

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 other 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**

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:

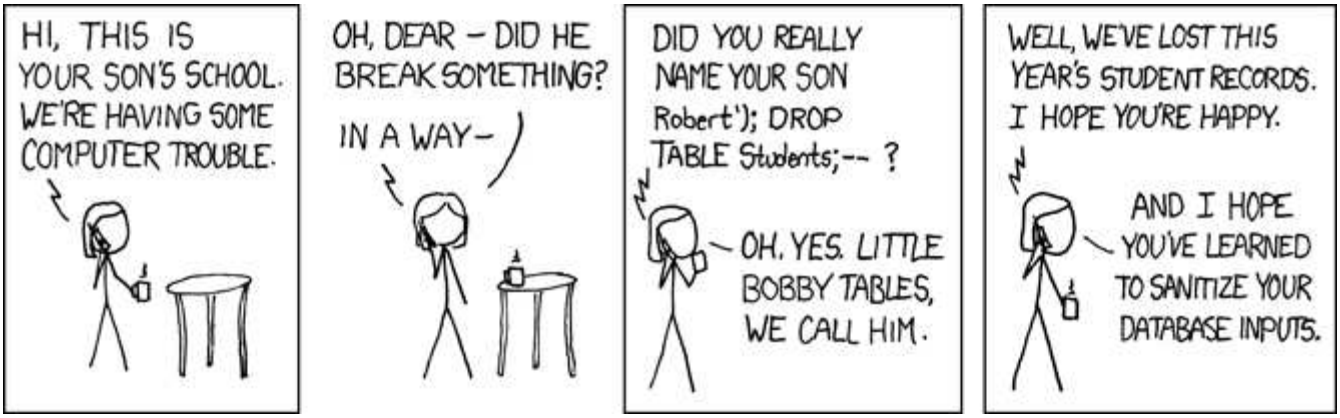
```php
$username = htmlspecialchars($_REQUEST["username"]);
```

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?

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

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