Web Security at Stanford

Feross Aboukhadijeh

Disclaimers

- I’m not an expert
- Web developers in the room?
  - If you’re a web developer, you might already know most of this stuff
- My goal: A few people learn something new

Web Security

- It’s very hard to get right
- We hear about websites getting hacked all the time in the news
- Reasons?
  - One scripting/programming language is embedded inside another
  - Web programmers are often self-taught, not aware of security implications of their code
  - Out of date server software is vulnerable to published, well-know exploits.
  - Bad language design

What is SQL injection?

- Incorrectly filtered escape characters

```
statement = ”SELECT * FROM users WHERE
  name = '” + userName + ”';”
```

What is SQL injection?

- Incorrectly filtered escape characters

```
statement = "SELECT * FROM users WHERE
  name = '" + userName + "';"
```

- What if my username is:

```
a' or 't'='t
```

What is SQL injection?

- Incorrectly filtered escape characters

```
statement = ”SELECT * FROM users WHERE
  name = ’” + userName + ”’;”
```

- What if my username is:

```
a’ or ’t’=’t
```

- Then the final query becomes:

```
SELECT * FROM users WHERE name = ’a’ OR
  ’t’=’t’;
```

What is SQL injection?

- Incorrectly filtered escape characters

```
statement = ”SELECT * FROM users WHERE
  name = '” + userName + ”';”
```

- What if my username is:

```
a';DROP TABLE users;
```

What is SQL injection?

- Incorrectly filtered escape characters

```
statement = ”SELECT * FROM users WHERE
  name = '” + userName + ”';”
```

- What if my username is:

```
a';DROP TABLE users;
```

- Then the final query becomes:

```
SELECT * FROM users WHERE name =
  'a';DROP TABLE users;
```

Examples

- In December 2009, an attacker breached a RockYou! plaintext database containing the unencrypted usernames and passwords of about 32 million users by using a SQL injection attack.

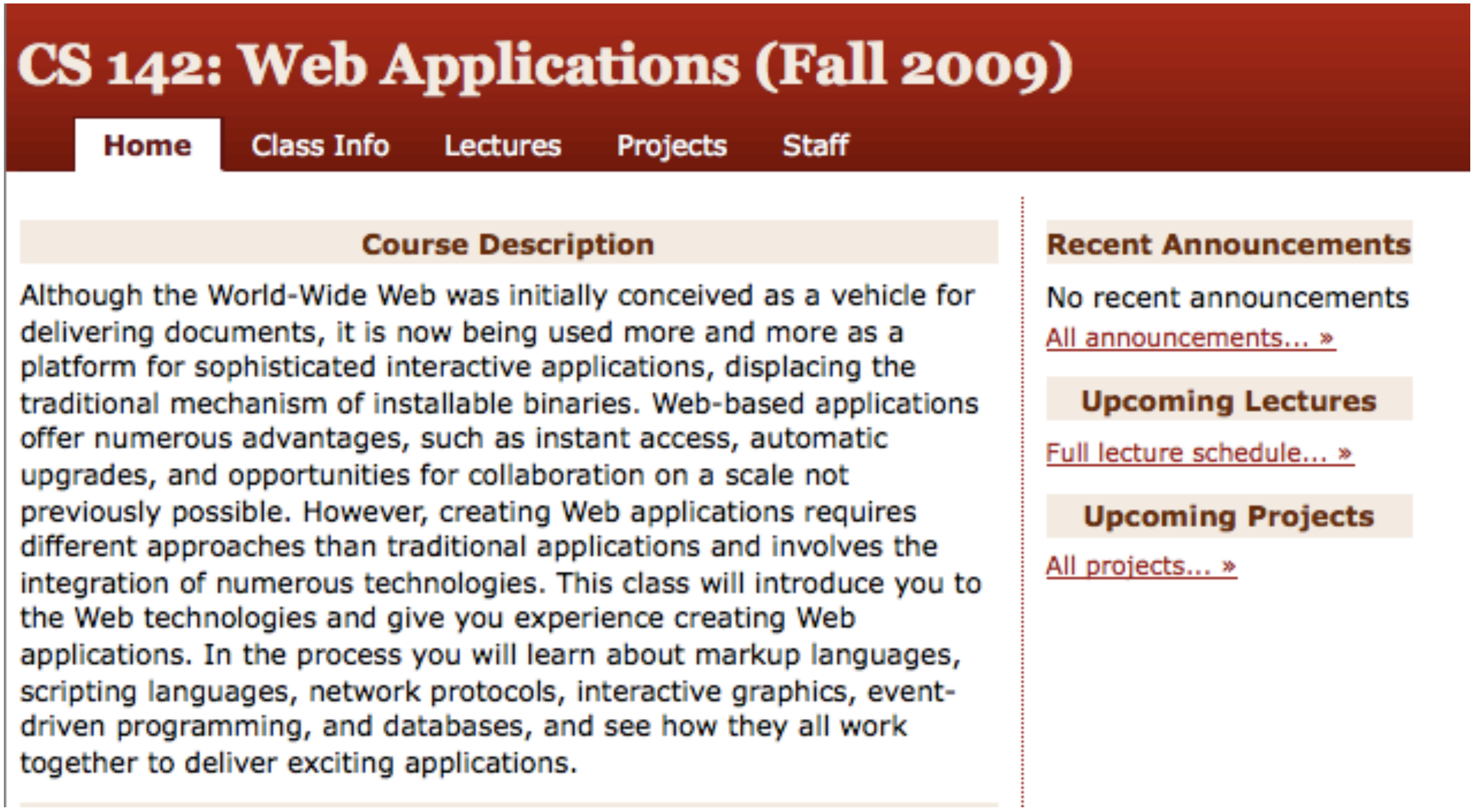
Source: NY Times

Examples

- On August 17, 2009, the United States Justice Department charged an American citizen Albert Gonzalez and two unnamed Russians with the theft of 130 million credit card numbers using a SQL injection attack. In reportedly "the biggest case of identity theft in American history", the man stole cards from a number of corporate victims after researching their payment processing systems. Among the companies hit were credit card processor Heartland Payment Systems, convenience store chain 7-Eleven, and supermarket chain Hannaford Brothers.

Source: BBC

My attack



CS 142: Web Applications (Fall 2009)

Course Description

Although the World-Wide Web was initially conceived as a vehicle for delivering documents, it is now being used more and more as a platform for sophisticated interactive applications, displacing the traditional mechanism of installable binaries. Web-based applications offer numerous advantages, such as instant access, automatic upgrades, and opportunities for collaboration on a scale not previously possible. However, creating Web applications requires different approaches than traditional applications and involves the integration of numerous technologies. This class will introduce you to the Web technologies and give you experience creating Web applications. In the process you will learn about markup languages, scripting languages, network protocols, interactive graphics, event-driven programming, and databases, and see how they all work together to deliver exciting applications.

Recent Announcements

No recent announcements

All announcements... »

Upcoming Lectures

Full lecture schedule... »

Upcoming Projects

All projects... »

My attack

Demo

My attack

- cs142/
  - cgi-bin/
    - lecture.php
    - lectures/
      - html.php
      - javascript.php
      - dom.php
      - …

My attack

- This URL:
  - http://www.stanford.edu/class/cs142/cgi-bin/lecture.php?topic=html
- Causes this line to get executed:
  - require_once(‘lectures/’ . $GET[‘topic’] . ‘.php’);

My attack

- This URL:
  - http://www.stanford.edu/class/cs142/cgi-bin/lecture.php?topic=html
- Causes this line to get executed:
  - require_once('lectures/' . $GET['topic'] . '.php');
- Where's the problem?
  - User-supplied data is executed as part of the PHP statement, without being cleaned first.
  - Always clean user-supplied data!

My attack

- I visit the URL:
  - http://www.stanford.edu/class/cs142/cgi-bin/lecture.php?topic=**../../../users/f/e/feross/malicious_file**
- Causes this line to get executed:
  - require_once('lectures/**../../../users/f/e/feross/malicious_file**.php');

My attack

- I visit the URL:
  - http://www.stanford.edu/class/cs142/cgi-bin/lecture.php?topic=**../../../users/f/e/feross/malicious_file**
- Causes this line to get executed:
  - require_once('lectures/**../../../users/f/e/feross/malicious_file**.php');
- Limitations
  - Required to be a file on the Stanford AFS network

My attack

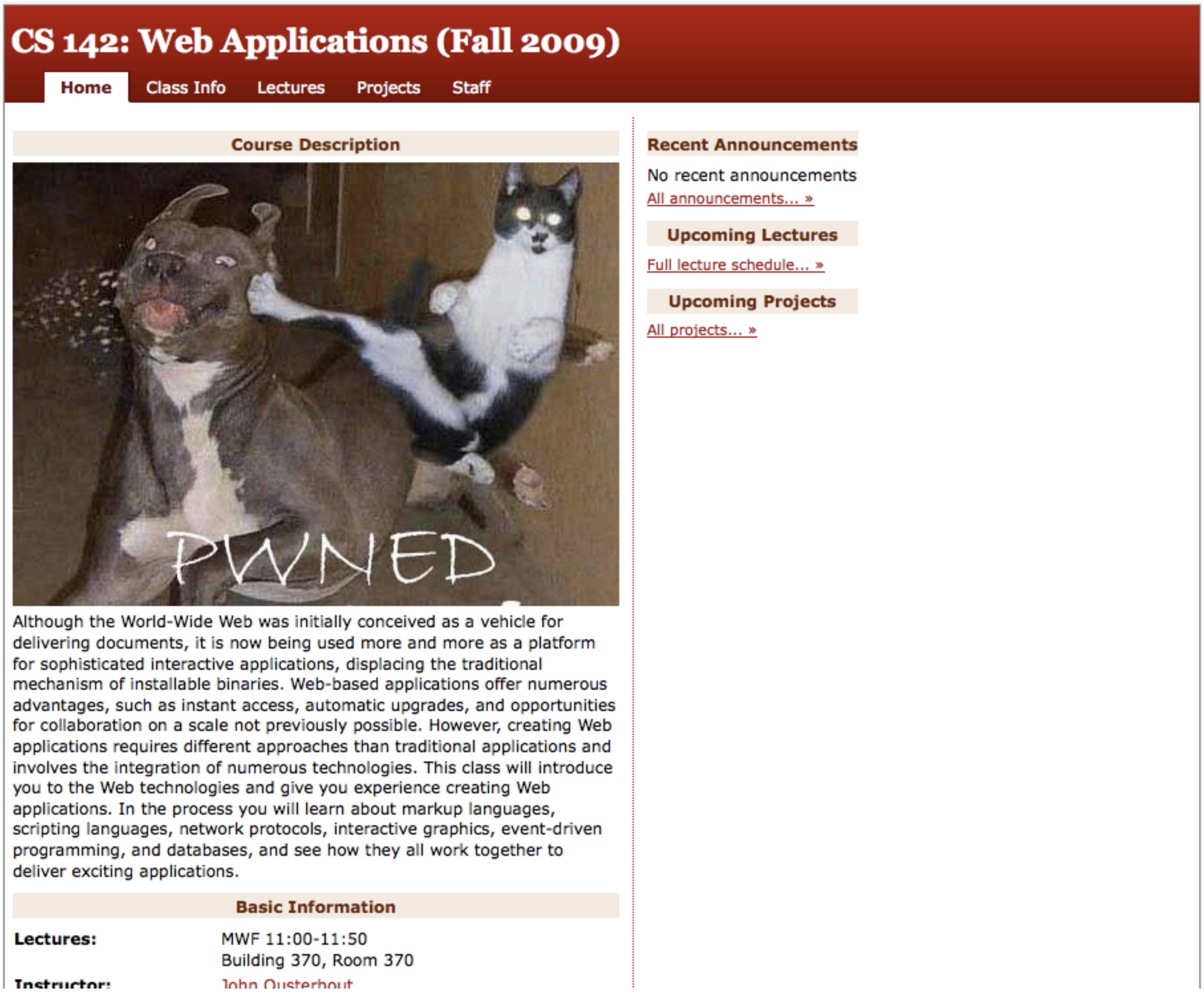
- Now whatever I put into **malicious_file.php** gets executed with the full permissions of the **cs142** user.
  - PHP allows you to do many fun things:
    - View, create, modify, delete files.
  - See all the other student's HW submissions
  - Redirect the page to anywhere on the Internet
  - Phishing?

My attack

- As described, this attack actually doesn’t work.
  - Who can guess why not?

My attack

- As described, this attack actually doesn't work.
  - Who can guess why not?
- **cs142** user does not have access to the files in my AFS space, but that's easy to fix
  - `fs setacl /afs/ir/users/f/e/feross cs142 rl`



CS 142: Web Applications (Fall 2009)

Course Description

Although the World-Wide Web was initially conceived as a vehicle for delivering documents, it is now being used more and more as a platform for sophisticated interactive applications, displacing the traditional mechanism of installable binaries. Web-based applications offer numerous advantages, such as instant access, automatic upgrades, and opportunities for collaboration on a scale not previously possible. However, creating Web applications requires different approaches than traditional applications and involves the integration of numerous technologies. This class will introduce you to the Web technologies and give you experience creating Web applications. In the process you will learn about markup languages, scripting languages, network protocols, interactive graphics, event-driven programming, and databases, and see how they all work together to deliver exciting applications.

Basic Information

Lectures:	MWF 11:00-11:50 Building 370, Room 370
Instructor:	John Ousterhout

Recent Announcements

No recent announcements

All announcements... »

Upcoming Lectures

Full lecture schedule... »

Upcoming Projects

All projects... »

Hotel Internet example

http://server/purchase.php?price=12.99&item=1234

Hotel Internet example

http://server/purchase.php?price=12.99&item=1234

http://server/purchase.php?price=**-50.00**&item=1234

ASSU Room Reservation System Vulnerability

- If time permits

ASSU Room Reservation System Vulnerability

- If time permits
- Code injection
  - HTML that you write is user-editable.
  - Even drop-down forms submitted by users should be sanitized before being used in your SQL queries.
    - Manually submit a malicious GET/POST request
    - Use Firebug to get the same effect

Things used to be real bad.

```
<?php
 if ($pass == "hello") // user auth
  $auth = 1;
 ...
 if ($auth == 1)
  echo "some important information";
?>
```

Things used to be real bad.

```
<?php
 if ($pass == "hello") // user auth
  $auth = 1;
 ...
 if ($auth == 1)
  echo "some important information";
?>
```

http://server/test.php?auth=1

Things used to be real bad.

http://server/test.php?auth=1

```
// PHP4
$auth // local variable

// PHP 5
$_GET['auth']
```

The future

- Web languages are getting better
- Most of the really obvious language problems don’t exist.

Patched!

- Both of the vulnerabilities I talked about have been fixed
- Don’t get any ideas….

Scope of the problem

- Lots of other Stanford sites are likely vulnerable
  - Hack away!
  - Be careful. Don’t do anything bad.

- White Hat Hacking Event (event where we hunt for security holes in Stanford-hosted websites and/or open source projects)
  - Are people interested?

Thanks!

Questions?