CMSC389R

Web I

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

recap

Any questions so far?

agenda

- Background
  - HTTP
  - HTTP requests (GET/POST)
  - Cookies, sessions, etc
- Common vulnerabilities
  - Cross-site scripting (XSS)
  - SQL injection (SQLi)
- In class challenge

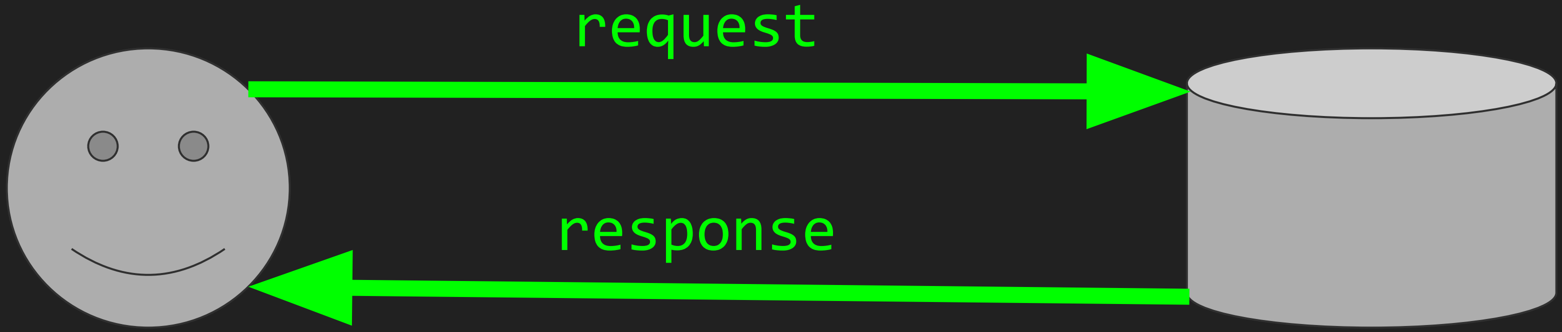
HTTP

- Hypertext Transport Protocol
    - Usually ports 80 (HTTP), 443 (HTTPS)
    - Stateless by design
        - Stateful by usage...
            - Think cookies & sessions!
    - Built on top of TCP
- Server-side code understands HTTP and responds to requests through this protocol

Basics of services

- What are web services built with these days?
    - Client-side
        - CSS/HTML/Javascript/etc…
    - Server-side
        - Yes, PHP is still actively used
        - Javascript/Python/Ruby/etc…
    - Databases
        - SQL, PostgreSQL, MongoDB, etc…

web basics



- browser
- curl
- wget
- ...

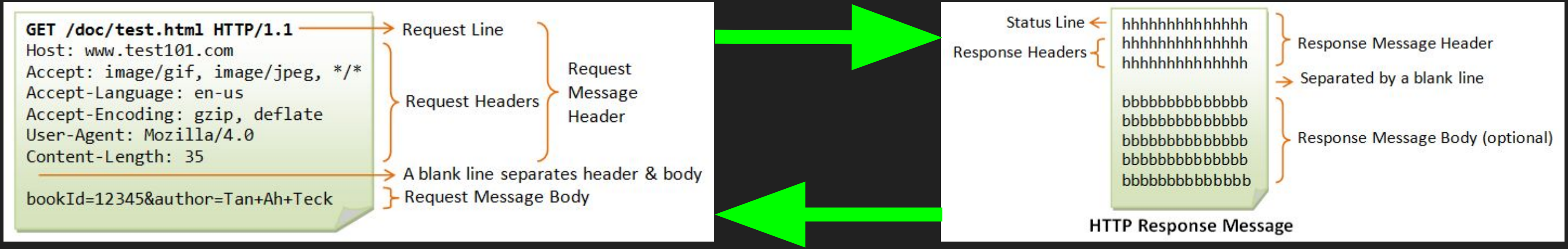
- website
- other server resources

basics of requests/responses

- When a user triggers an action on the front-end
  - Typically send a request (GET/POST/PUT/...) to the server
  - Server receives request
    - Handles it (data processing/...)
    - Responds
- Front-end handles server's response
  - Browser renders DOM

HTTP request basics

GET and POST*



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

```
hhhhhhhhhhhhh
hhhhhhhhhhhhh
hhhhhhhhhhhhh

bbbbbbbbbbbbb
bbbbbbbbbbbbb
bbbbbbbbbbbbb
bbbbbbbbbbbbb
bbbbbbbbbbbbb
```

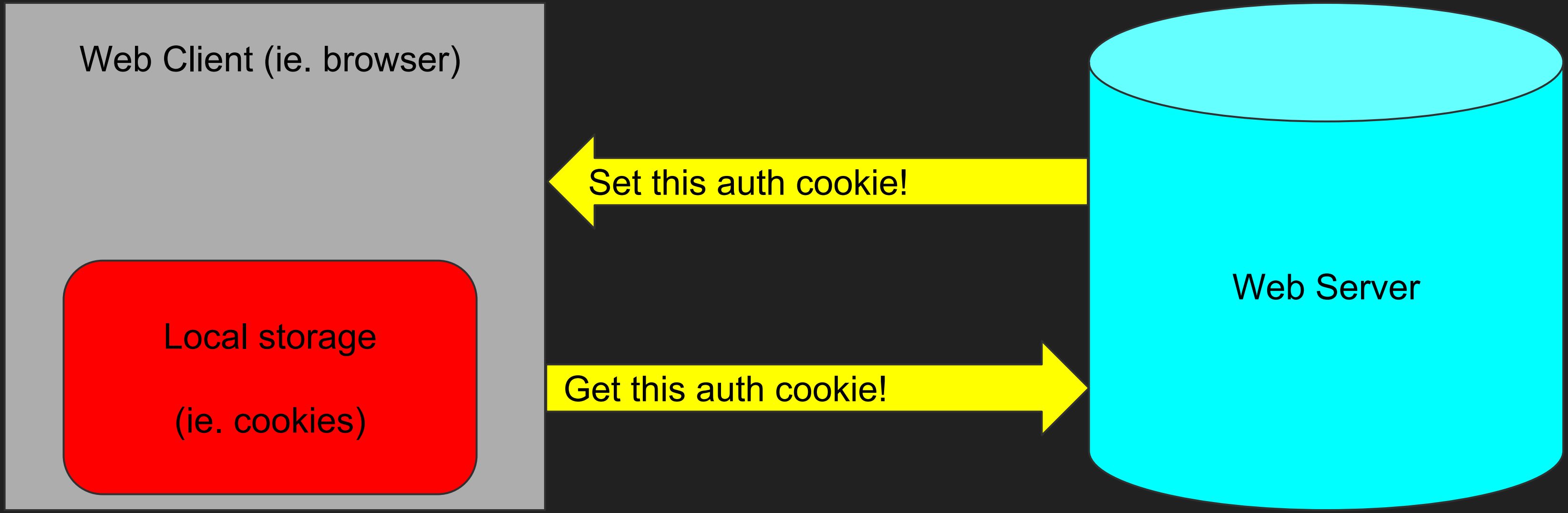
*there are others, but we'll focus on these

Cookies

- Piece of data stored client-side
  - Typically passed around in sessions
  - Completely r/w by the client
- Can be dangerous if not used correctly by server!
  - Can be modified in the browser

```
document.cookie="keyofcookie=valueofcookie"
```

Cookies


Web Client (ie. browser)
Local storage
(ie. cookies)
Set this auth cookie!
Get this auth cookie!
Web Server

Challenge!

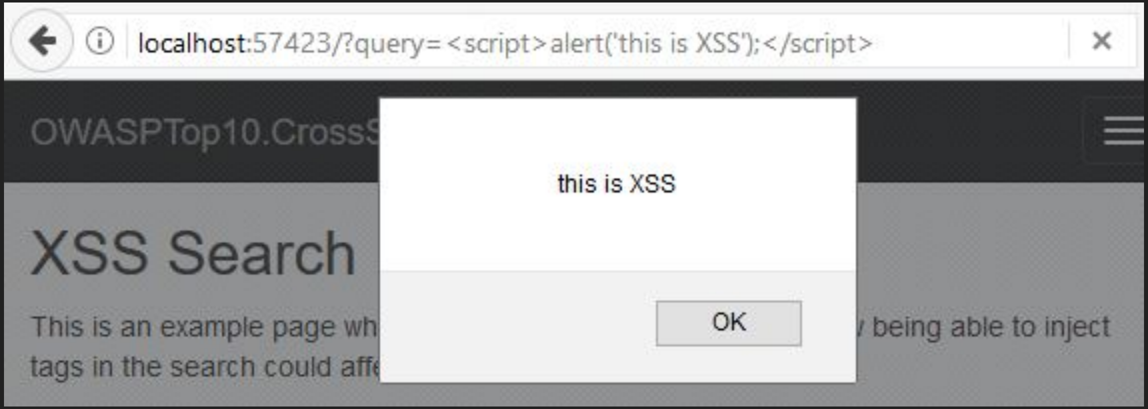
http://68.183.48.170:2600

Injection Attacks

cross site scripting (xss)

- Attacker sends malicious code rendered on the victim’s browser
    - Stored: attacker forces malicious code to be stored on database
        - ie) user sets username to injection code; rendered each time victim visits profile
    - Reflected: injected script is reflected off of server to victim
        - Typically sent via email/links/...

cross site scripting (xss)


```
localhost:57423/?query=<script>alert('this is XSS');</script>
```
OWASPTop10.CrossS

this is XSS

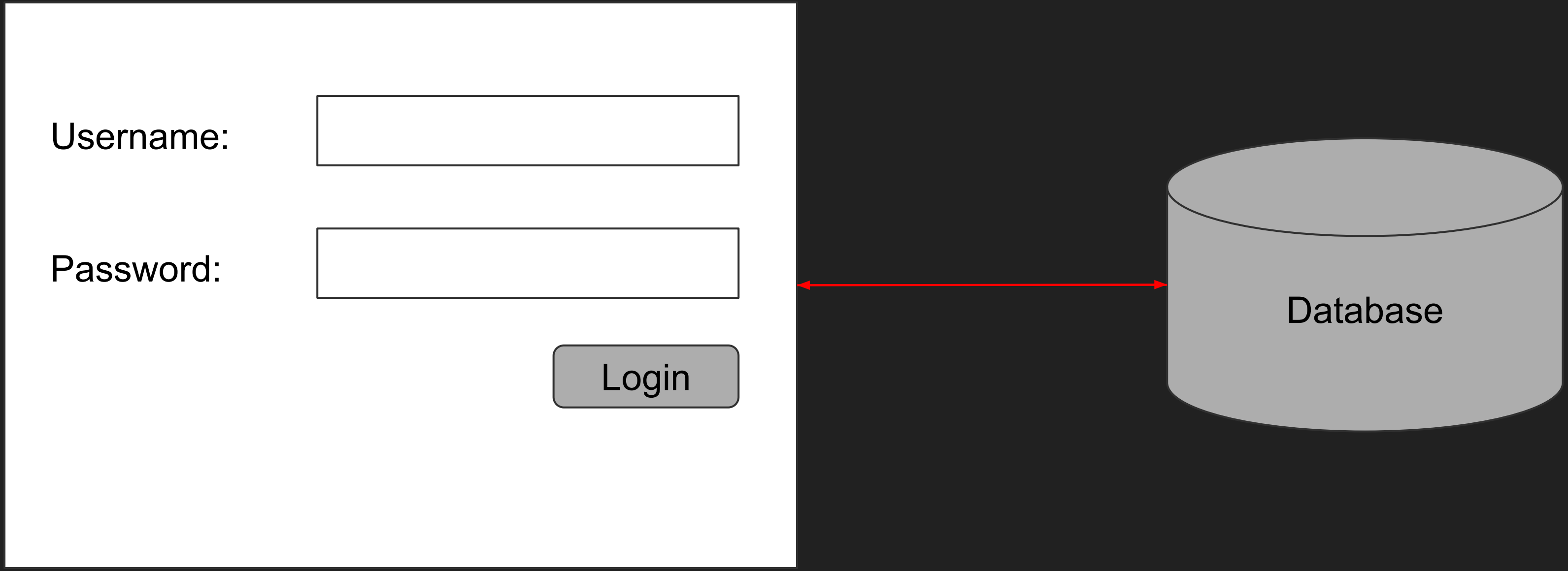
XSS Search

This is an example page wh / being able to inject tags in the search could affe

sql injection

- Website utilizes SQL database
  - Does not sanitize input
  - Query is interpreted as code rather than data
  - Mitigated with prepared statements
- Potentially leads to:
  - Leaking tables
  - Deleting tables
  - Command execution

sql injection



sql (primer)

Table: users

user	password	last_login_date
admin	password1234	1542392929
michael	this_is_a_bad_password	1542392920

```
SELECT password from users where user = ‘admin’;  → password1234

INSERT INTO users values(‘new_user’, ‘terps’, ‘1542392927’);

SELECT * from users; -- this is a comment
```

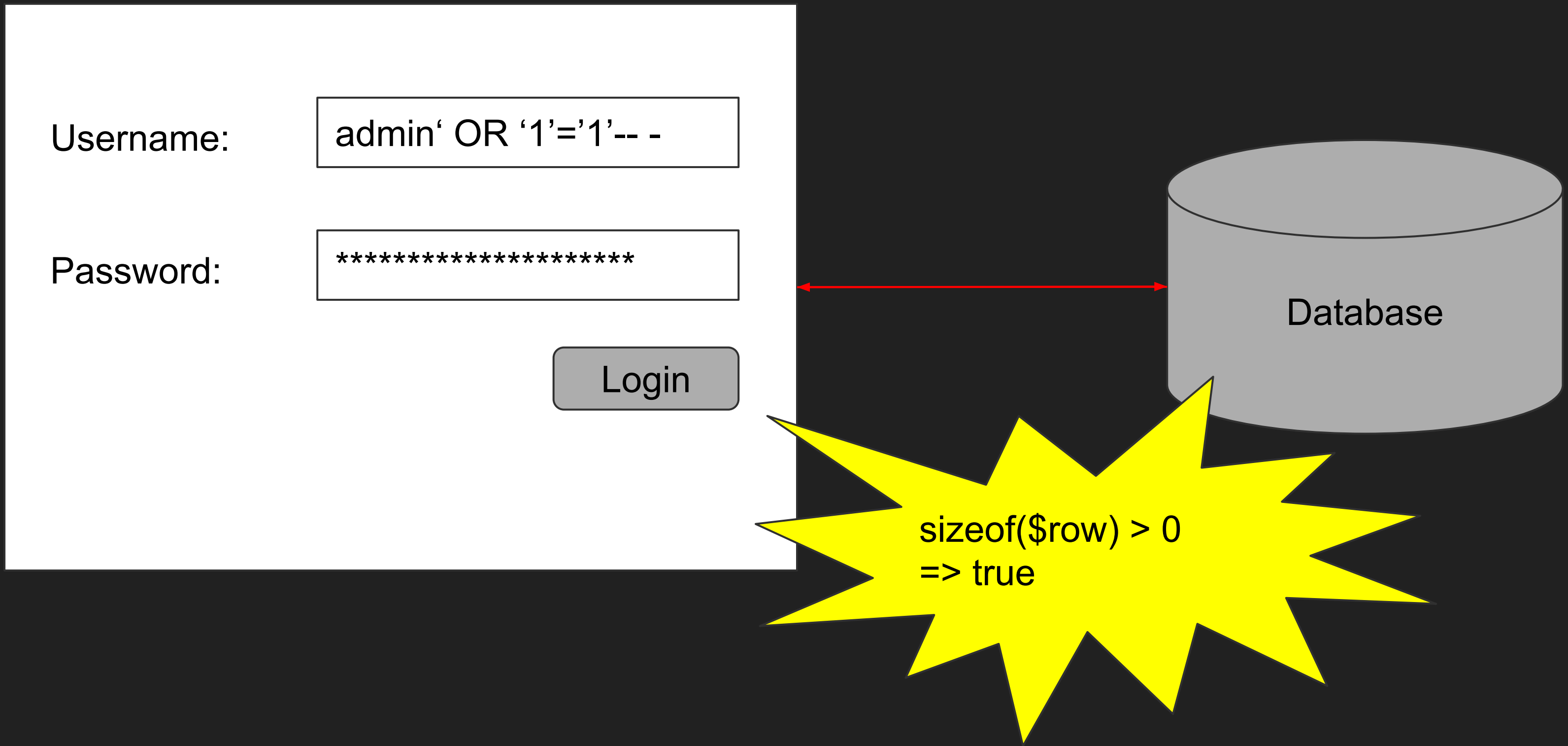
sql injection

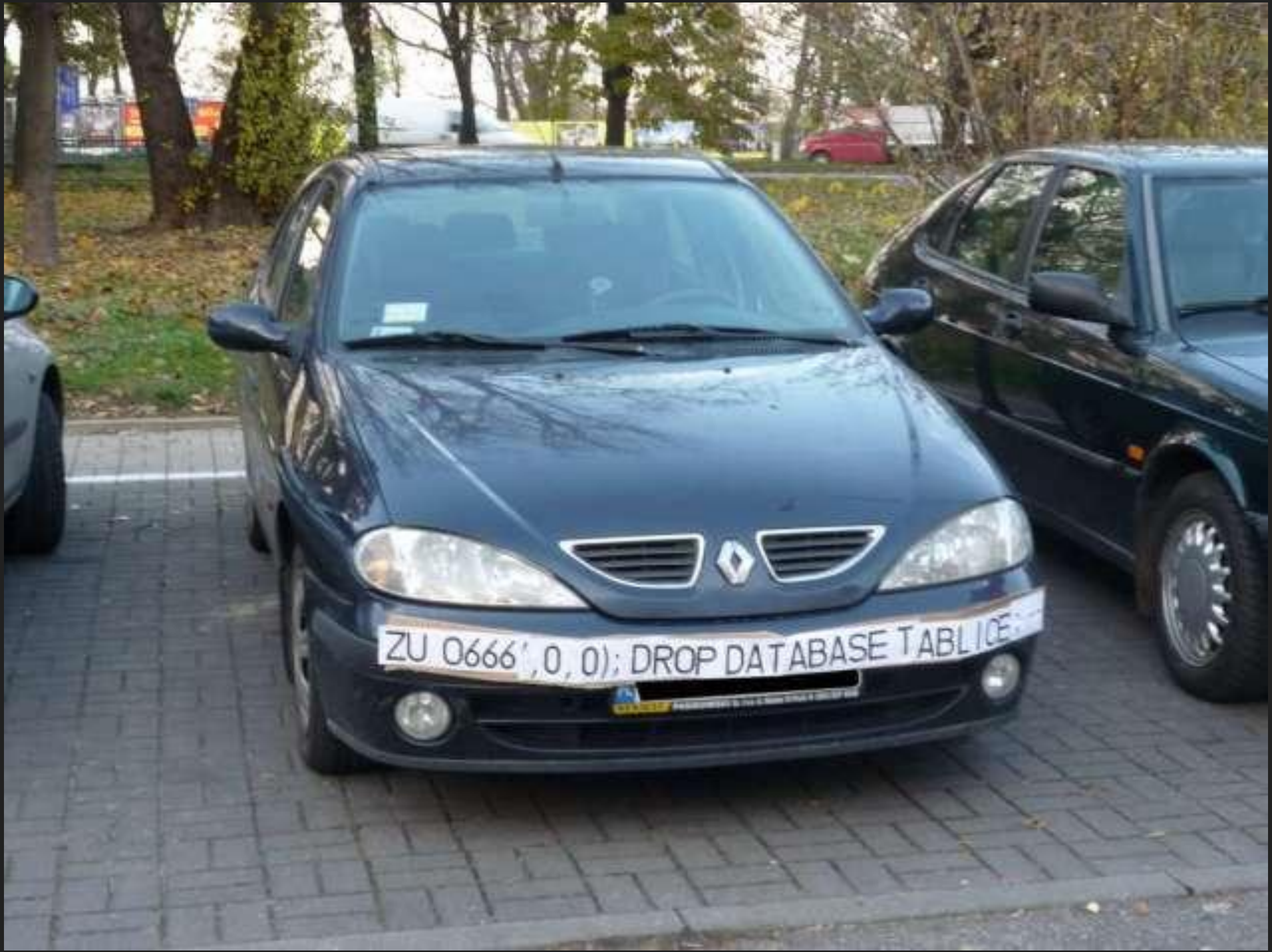
```
function can_access_feature($current_user) {
     global $db_link;

     $db_link = mysqli_connect('localhost', 'dbuser', 'dbpassword', 'dbname');
     $username = $_POST['username'];
     $password = $_POST['password'];

     $res = mysqli_query($db_link, "SELECT * FROM users WHERE username = '".
$username . "' AND password = '" . $password. "';");

     $row = mysqli_fetch_array($res);
     if (sizeof($row) > 0) {
         return true;
     } else {
         return false;
     }
}
```

sql injection

```
function can_access_feature($current_user) {
     global $db_link;

     $db_link = mysqli_connect('localhost', 'dbuser', 'dbpassword', 'dbname');
     $username = $_POST['username'];
     $password = $_POST['password'];

     $res = mysqli_query($db_link, "SELECT * FROM users WHERE username = '".
$username . "' AND password = '" . $password. "';");

     $row = mysqli_fetch_array($res);
     if (sizeof($row) > 0) {
         return true;
     } else {
         return false;
     }
}
```

sql injection




ZU 0666',0,0); DROP DATABASE TABLICE;--
[redacted]

resources

- Natas OverTheWire
- JuiceShop
- Gruyere
- Ringzer0team
- OWASP Top 10

homework #11

NO HOMEWORK THIS WEEK

GO ENJOY THANKSGIVING!