Web Security

Today

- Web architecture
  - Basics of web security

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

What is the Web?

- A platform for deploying applications, *portably* and *securely*

Assignment Project Exam Help
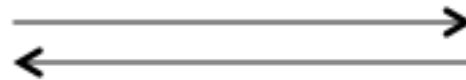
https://powcoder.com

Add WeChat powcoder

client   server

3

Web security: two sides

- Web browser: (client side)
  - Interacts with the user
  - Fetches and renders pages from the server
  - Worry about user's own data, malware, keyloggers, …

- Web application: (server side)
  - Runs at web site: banks, e-merchants, blogs
  - Written in PHP, ASP, JSP, Python, Ruby, Node.js, …
  - Decides which page to serve to which users/requests (authorized users, dynamic content, etc)
  - Worry about all users' data, targeted attacks, …

A historical perspective

- The web is an example of "bolt-on security"
- Originally, the web was invented to allow physicists to share their research papers
  - Only textual web pages + links to other pages; no security model to speak of
- Then we added embedded images
  - Crucial decision: a page can embed images loaded from another web server
- Then, Javascript, dynamic HTML, AJAX, CSS, frames, audio, video, …
- Today, a web site is a distributed application

URLs

- Global identifiers of network-retrievable documents

- Example:

Assignment Project Exam Help

https://powcoder.com

http://ecen4133.org:80/tmp/test?foo=1337#top

Add WeChat powcoder

Protocol

host

Path

Query

Fragment

Are URLs case-sensitive?

HTML

- Hypertext markup language (HTML)
  - Describes the content and formatting of Web pages
  - Rendered within browser window
- HTML features
  - Static document description language
  - Supports linking to other pages and embedding images by reference
  - User input sent to server via forms
- HTML extensions
  - Additional media content (e.g., PDF, video) supported through plugins
  - Embedding programs in supported languages (e.g., JavaScript, Java) provides dynamic content that interacts with the user, modifies the browser user interface, and can access the client computer environment

HTTP protocol

- HTTP is
  - widely used
  - Simple
  - Stateless

8

HTTP Protocol

Method   File   HTTP version

GET /index.html HTTP/1.1

Headers

Host: www.example.com

HTTP Request

Browser

HTTP Reply

Web Server

HTTP version   Status code   Reason phrase

HTTP/1.1 200 OK
Date: Tue, 14 Feb 2012
Server: Apache/1.3.3.7
Last-Modified: Mon, 13 Feb 2012
Content-Length: 438
Set-Cookie: ...
Content-Type: text/html; charset=UTF-8

Headers

```
<html>
 <head>
  <title> Hello World! </title>
 </head>
 <body>
  <p> Hello World! </p>
 </body>
</html>
```

Cookies

Data

9
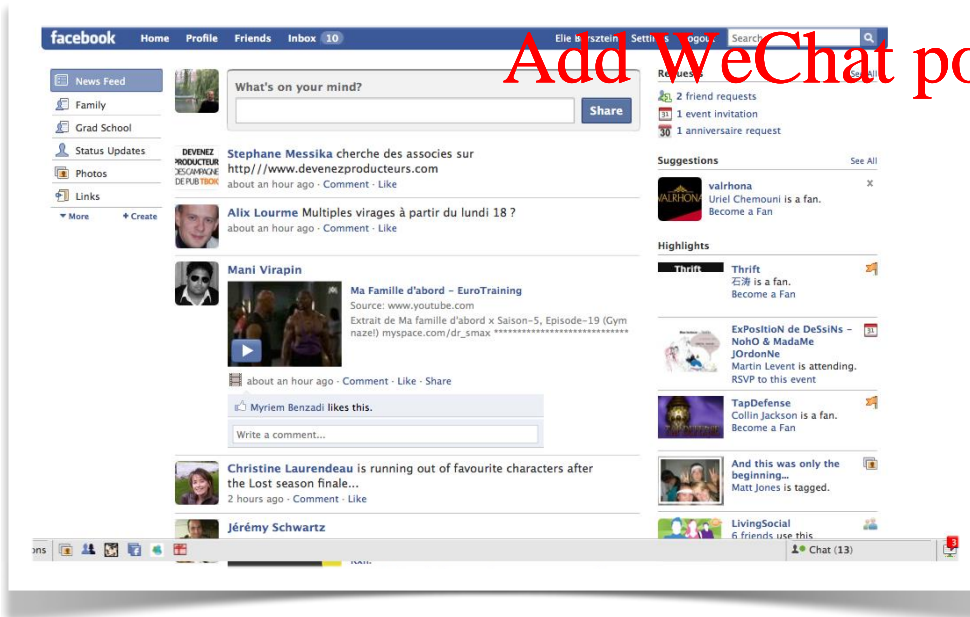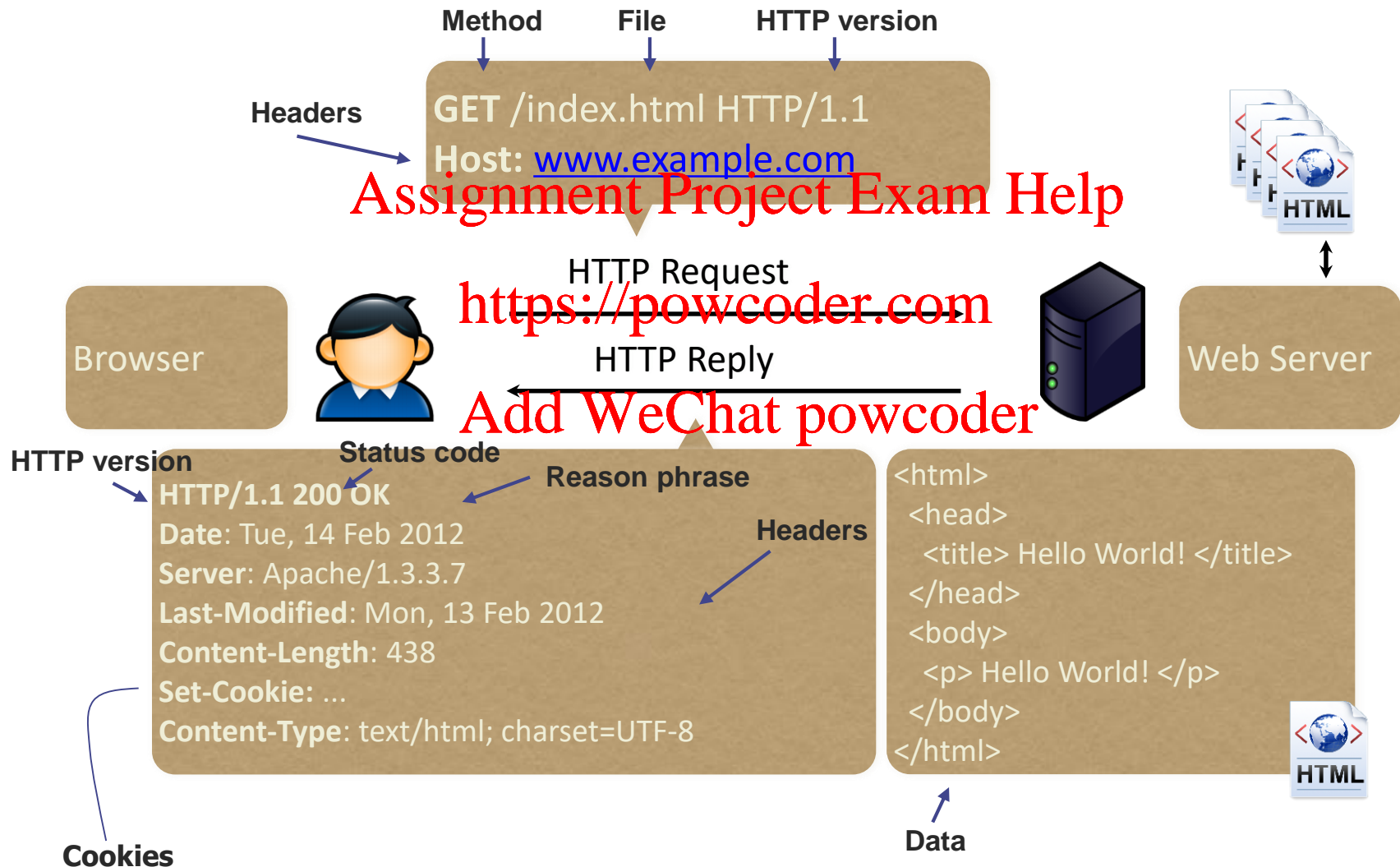
HTTP GET request

- Used to fetch resources

- Shouldn't change state on the server

```
GET /cat.jpg HTTP/1.1
Host: catpictures.net
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/56.0.2924.76 Safari/537.36
Accept: text/html,application/xhtml+xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: en-US,en;q=0.8
```

10

HTTP POST request

- Used to update state on the server

- Clients can send/upload files/data

```
POST /register HTTP/1.1
Host: catpictures.net
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/56.0.2924.76 Safari/537.36
Accept: text/html,application/xhtml+xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: en-US,en;q=0.8
Content-Length: 20

User=bob&Pass=abc123
```

11

HTML Basics

- HyperText Markup Language
  - Nested "tag" structure
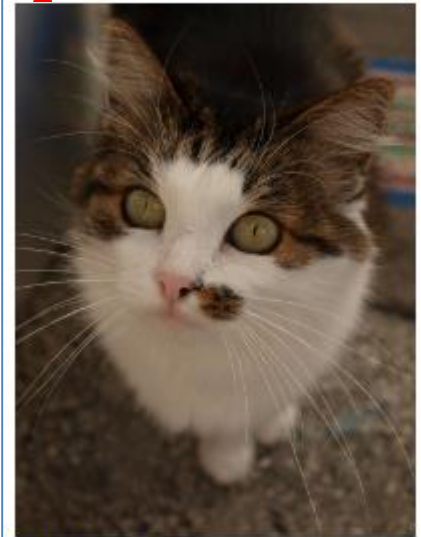
```
<html>
  <head>
    <title>Cats!</title>
  </head>
  <body>
    <h1>Look, a cat!</h1>
    <span>
      <img src="cat.jpg" width="200px"/>
      <br/>
      <a href="cats.html">
        Click here for more cats!
      </a>
    </span>
  </body>
</html>
```


Look, a cat!

Click here for more cats!

12

HTML, CSS, Javascript

- HTML for **structure**
  - What elements of a page are related?
  - What resources should be included?

- CSS (Cascading Style Sheet) for **style**
  - What fonts/colors/sizes/positions should elements be?

- Javascript for **dynamic content**
  - When a user clicks this, do that
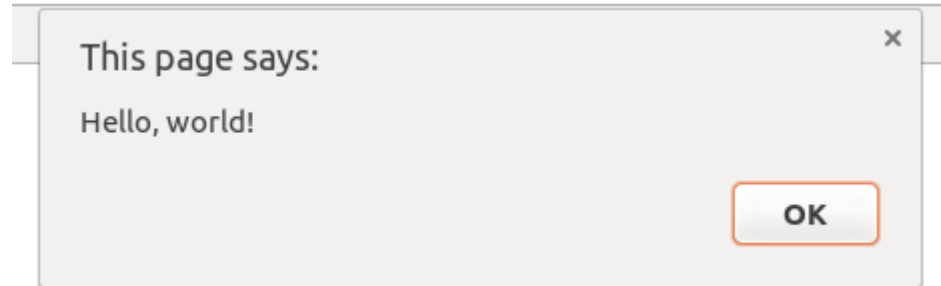 - *Here be dragons!*

13

Javascript

```
<html>
  <head>
    <script type="text/javascript">
      alert('Hello, World!');
    </script>
  </head>
  <body>
  </body>
</html>
```

This page says:

Hello, world!

OK

14

DOM Tree: Document Object Model

- "The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents."

Javascript

- Functional, imperative, and object-oriented

- Oh, and untyped. Good luck! ☺

```
function factorial(x) {
    var r = x;
    for (var i=1; i<x; i++) {
        r *= (x-i);
    }
    return r;
}

alert(factorial(10));

setTimeout(function () {
            alert(factorial(10));
        }, 1000);
```

Functional!

```javascript
// Assign unnamed (anonymous) functions to variables
var factorial = function (x) {
    var r = x;
    for (var i=1; i<x; i++) {
        r *= (x-i);
    }
    return r;
};

// Anonymous functions can be passed like a function pointer
setTimeout(function () {
            alert(factorial(10));
        }, 1000);

// You can even call anonymous functions!
(function (name) { alert('Hello, ' + name))('Alice');
```

Untyped weirdness!

```
var x = 'dog' + 5; // 'dog5'


x = '5' + 3;        // '53'
x = '5' – 3;        // 2
x = 'dog' – 3;      // NaN

// No need to memorize this, but just know JS is weird...
x = [];             // An (empty) array
x = ![];            // false
x = 0 + [];         // "0"
x = +[];            // 0
x = +!+[];          // 1  (because !0 == true and +true == 1)
// Can write any Javascript program using only 6 characters:
// JSFuck:      ()+[]!
```

Javascript accessing the DOM

```
<html>
  <span id="foo">
    <a href="prize.html">Click here, quick!</a>
  </span>
  <script>
    function too_late() {
        document.getElementById('foo').innerText = "TOO SLOW!";
    }
    setTimeout(too_late, 100);
  </script>
</html>
```

19

JQuery

```html
<html>
  <script src="jquery-3.1.0.min.js"></script>
  <span id="foo">
    <a href="prize.html">Click here to win!</a>
  </span>
  <script>
    function too_late() {
        $('#foo').innerText = "TOO SLOW!";
    }
    setTimeout(too_late, 100);
  </script>
</html>
```

20

AJAX (w/ JQuery)

```html
<html>
  <script src="jquery-3.1.0.min.js"></script>
  <span id="foo">Loading the weather...</span>
  <script>
    $(function() {
        // This function will be called on DOM load

        // $.get(url, cb) makes an asynchronous retrieval of
        // the provided URL, and calls the second argument
        $.get("https://site.com/weather", function (data) {
                // This function is called with the result of
                // loading the URL
                $('#foo').html(data);
        });
    });
  </script>
</html>
```

Web Review | HTTP

GET / HTTP/1.1
Host: gmail.com

http://gmail.com/
says:

Hi!

HTTP/1.1 200 OK
...
<html>
<head>

<script>alert('Hi!')</script>
</head>

gmail.com

GET /img.png HTTP/1.1
Host: gmail.com

HTTP/1.1 200 OK
...
<89>PNG^M ...

Web Review | Cookies

POST /login HTTP/1.1
Host: gmail.com

user=alice&pass=s3cre7

Assignment Project Exam Help

https://powcoder.com

HTTP/1.1 200 OK

Server: gws

Set-Cookie: foo="bar"

Set-Cookie: token="8kFmCe…"

Add WeChat powcoder

<html>

…

gmail.com

Ah, it's alice!

GET / HTTP/1.1
Host: gmail.com
Cookie: foo="bar"; token="8k…"

Cookies

- Cookies are a small bit of information stored on a computer associated with a specific server
  - When you access a specific website, it might store information as a cookie
  - Every time you revisit that server, the cookie is re-sent to the server
  - Effectively used to hold state information over sessions
- Cookies can hold any type of information
  - Can also hold sensitive information
    - This includes passwords, credit card information, social security number, etc.
    - Session cookies, non-persistent cookies, persistent cookies
  - Almost every large website uses cookies

More on Cookies

- Cookies are stored on your computer and can be controlled
  - However, many sites require that you enable cookies in order to use the site
  - Their storage on your computer naturally lends itself to exploits (Think about how ActiveX could exploit cookies...)
  - You can (and probably should) clear your cookies on a regular basis
  - Most browsers will also have ways to turn off cookies, exclude certain sites from adding cookies, and accept only certain sites' cookies
- Cookies expire
  - The expiration is set by the sites' session by default, which is chosen by the server
  - This means that cookies will probably stick around for a while

25

"Evercookie"

- Cookies are just state servers store in your browser

- Where else can state be stored?

  – Local Storage (Javascript)

  – Image/resource cache

  – Flash local shared objects

  – Java storage

  – Others?

Third-party cookies

27

Third-party cookies

- External (third-parties) serve ads/trackers on sites you visit
- The trackers can tell what page you are visiting, and **correlate** this with other page visits your browser makes
- Trackers then sell info to advertisers (often in real time!), who use it to show you ads relevant to you

Third-party cookies

- Most browsers let you disable third-party cookies
  - Only send cookies to first-party sites, never to external/third-party sites

Allow sites to save and read cookie data (recommended)

Clear cookies and site data when you quit Chrome

Block third-party cookies
When on, sites can't use your browsing activity across different sites to personalize ads. Some sites may not work properly.
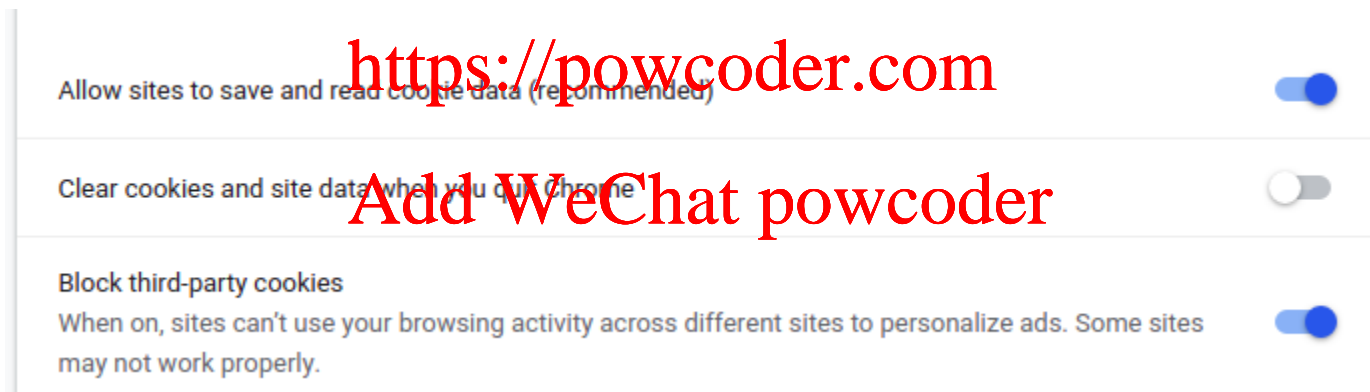
- Privacy extensions: AdBlock, Ghostery, PrivacyBadger, etc block trackers

Taking Care of Your Cookies

- Managing your cookies in Chrome:
  - Remove Cookie
  - Remove All Cookies
  - Displays information of individual cookies
  - Also tells names of cookies, which probably gives a good idea of what the cookie stores
    - i.e. amazon.com: session-id

Cookies and site data

Site	Locally stored data
blog.0x21.io	2 cookies, Local storage
meet.0x21.io	Local storage
_____.net	Channel ID
3druck.com	1 cookie
99percentinvisible.org	1 cookie
acm.org	2 cookies
dl.acm.org	5 cookies
www.acm.org	Local storage
_____yogi.org	1 cookie
code.activestate.com	1 cookie
helpx.adobe.com	Local storage
www.adobe.com	1 cookie, Local storage
adwordsbanned.net	1 cookie
agkn.com	2 cookies
airbnb.com	7 cookies
www.airbnb.com	1 cookie, Local storage

Remove all   Search cookies

Done

Security on the web

- Web sites should not be able to read or change files on my computer

- Web sites should not be able to learn what other websites I visit, or how I interact with them

- Web sites should not be able to cause me to interact with other unrelated websites

Security on the web

- Risk #1: we don't want a malicious site to be able to trash my files/programs on my computer

  Assignment Project Exam Help

  https://powcoder.com
  - Browsing to awesomevids.com (or evil.com) should not infect my computer with malware, Add WeChat powcoder read or write files on my computer, etc.

- Defense: Javascript is sandboxed;
  try to avoid security bugs in browser code;
  privilege separation; automatic updates; etc.

Security on the web

- Risk #2: we don't want a malicious site to be able to spy on or tamper with my information or interactions with other web sites

  – Browsing to evil.com should not let evil.com spy on my emails in Gmail or buy stuff with my Amazon account, even if I am logged in

- Defense: the **same-origin policy**

  – A security policy grafted on after-the-fact, and enforced by web browsers

  – Intuition: each web site is isolated from all others

Security on the web

- Risk #3: we want data stored on a web server to be protected from unauthorized access

  - More on this in the web project!

  - Next week: XSS, CSRF, SQL injection...
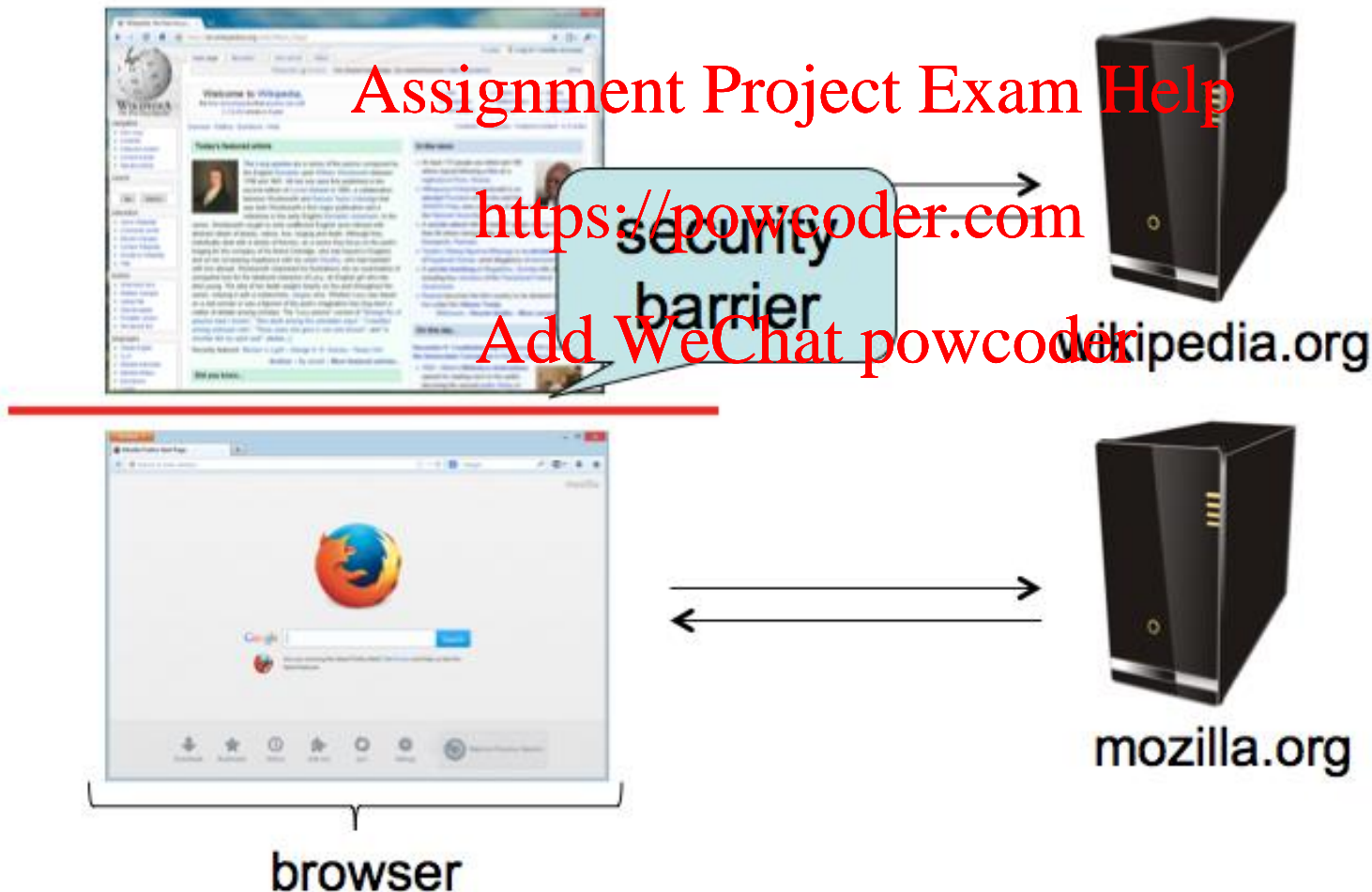
- Defense: server-side security

Same-origin policy

- Each site is isolated from all others

security barrier

wikipedia.org

browser

mozilla.org

35

Same-origin policy

- Multiple pages from same site aren't isolated

36
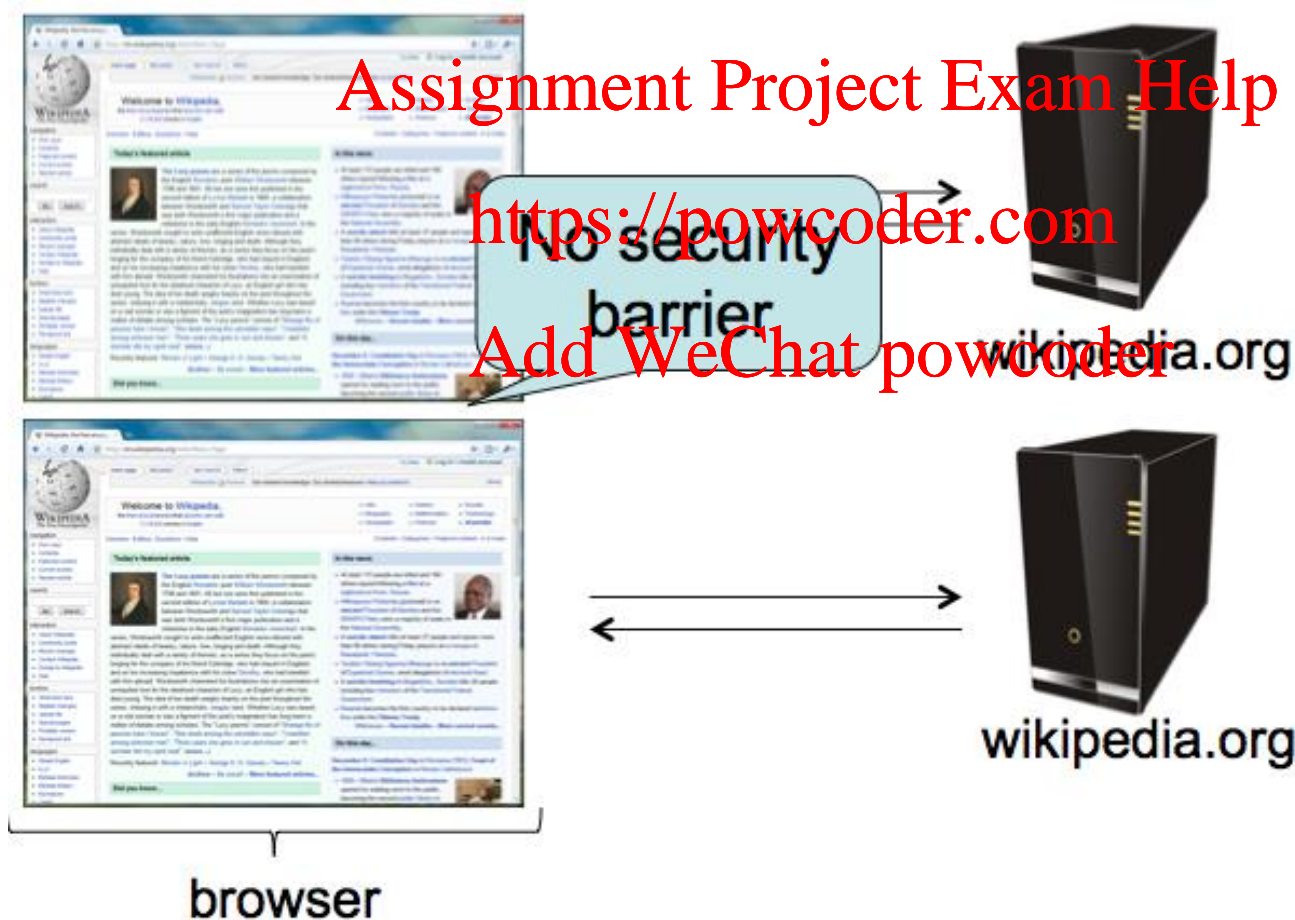
Same-origin policy

- Granularity of protection: the *origin*

- Origin = protocol + hostname (+ port)

Assignment Project Exam Help

http://coolsite.com/tools/info.html

https://powcoder.com

protocol

Add WeChat powcoder

hostname

- Javascript on one page can read, change, and interact freely with all other pages from the same origin

Same-origin policy

- The origin of a page (frame, image, …) is derived from the URL it was loaded from

Assignment Project Exam Help

- Special case: Javascript runs with the origin of the page that loaded it

https://powcoder.com

Add WeChat powcoder

Confining the Power of JavaScript Scripts

- Given all that power, browsers need to make sure JS scripts don't abuse it

- For example, don't want a script sent from hackerz.com web server to read cookies belonging to bank.com ...

- ... or alter layout of a bank.com web page

- ... or read keystrokes typed by user while focus is on a bank.com page!

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Same-origin policy

- Browsers provide isolation for JS scripts via the Same Origin Policy (**SOP**)
- Simple version:
  - Browser associates web page elements (layout, cookies, events) with a given **origin** ≈ web server that provided the page/cookies in the first place
    - Identity of web server is in terms of its hostname, e.g., bank.com
- SOP *= only scripts received from a web page's origin have access to page's elements*
- **XSS: Subverting the Same Origin Policy**