Web Security 1

Prof. Raluca Ada Popa

Oct 16, 2017

Some content adapted from materials by David Wagner or Dan Boneh

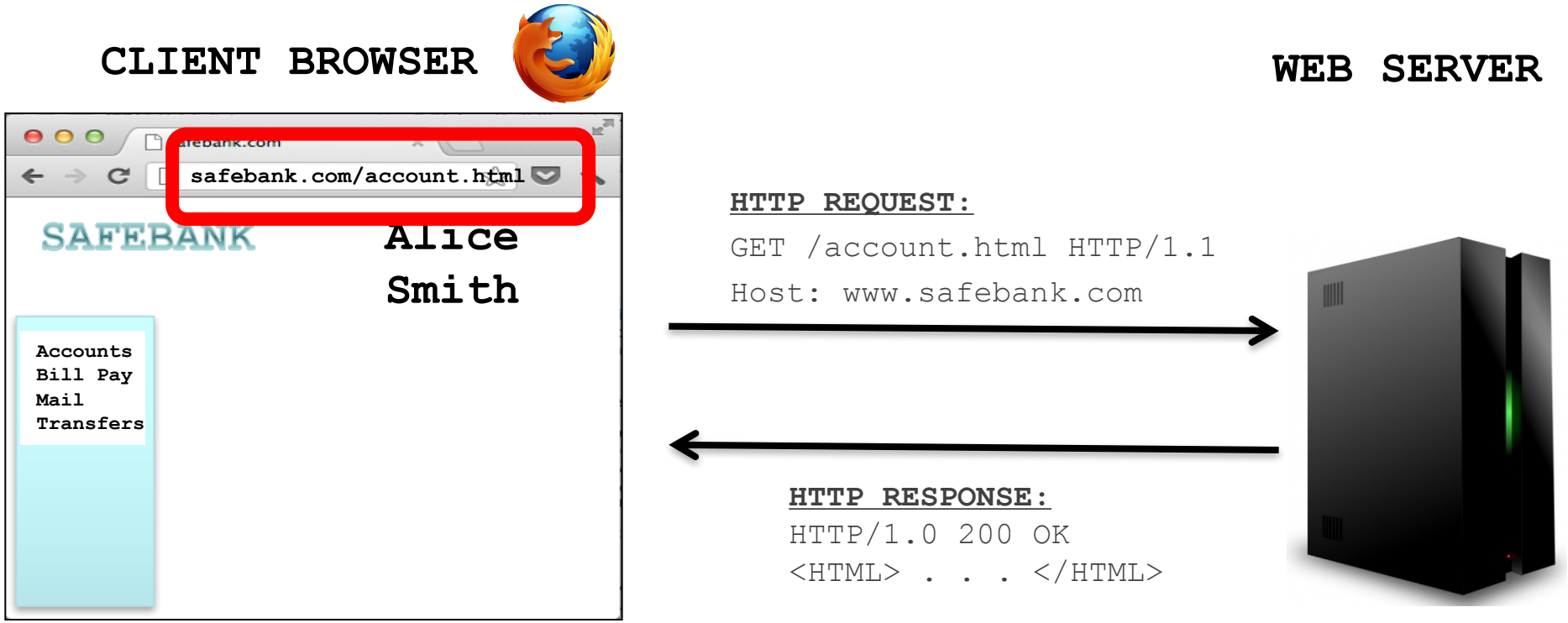
Today

- We need to cover same-origin policy, cookie policy, CSRF and XSS, but do not need to cover web injection
- Scribe: Dayeol
- Presenter: Rohan, Michael

HTTP

(Hypertext Transfer Protocol)

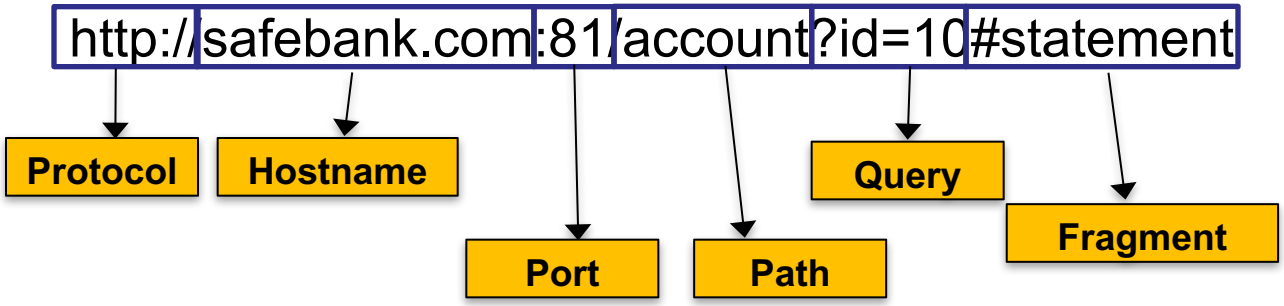
A common data communication protocol on the web



URLs

Global identifiers of network-retrievable resources

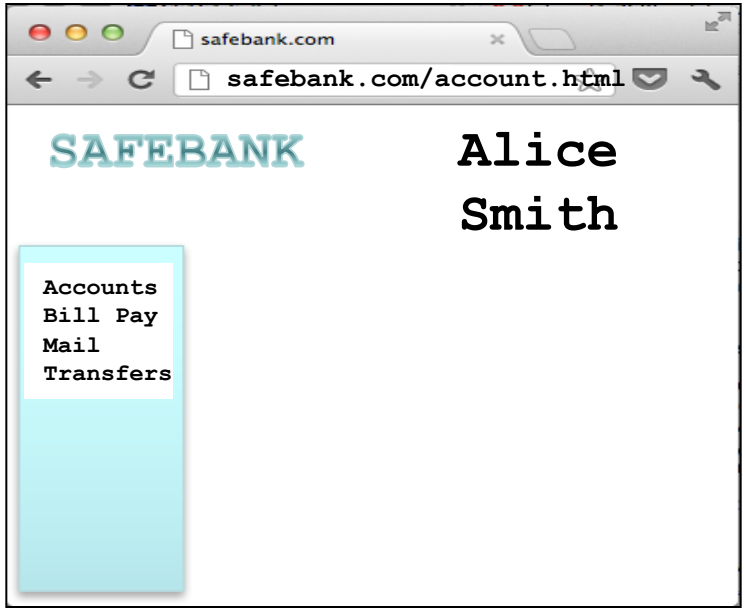
Example:



HTTP

CLIENT BROWSER

WEB SERVER



HTTP REQUEST:

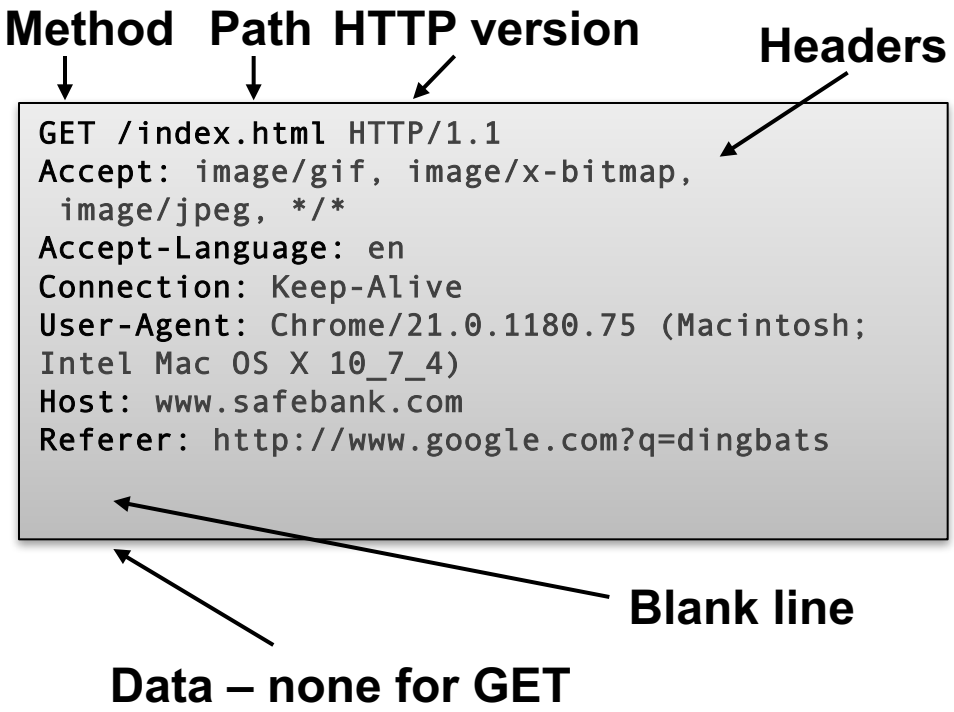
```
GET /account.html HTTP/1.1
Host: www.safebank.com
```

HTTP RESPONSE:

```
HTTP/1.0 200 OK
<HTML> . . . </HTML>
```

HTTP Request

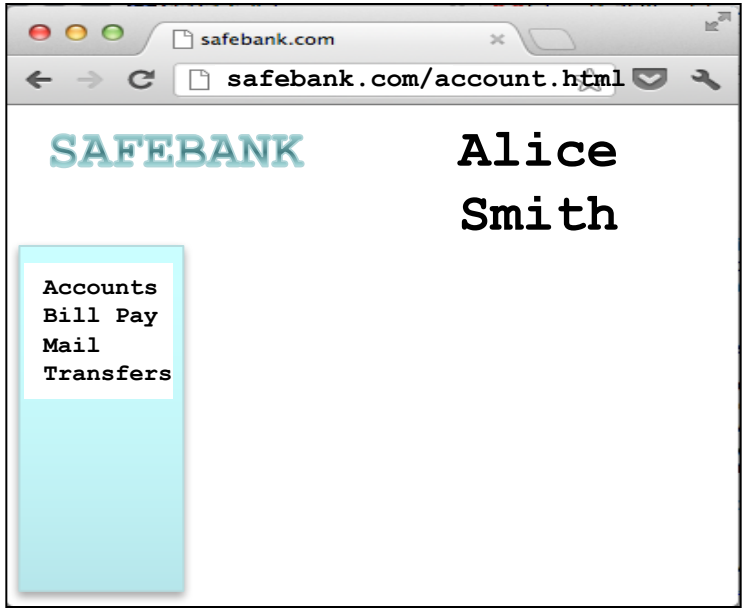
GET: no side effect
POST: possible side effect



HTTP

CLIENT BROWSER

WEB SERVER



HTTP REQUEST:

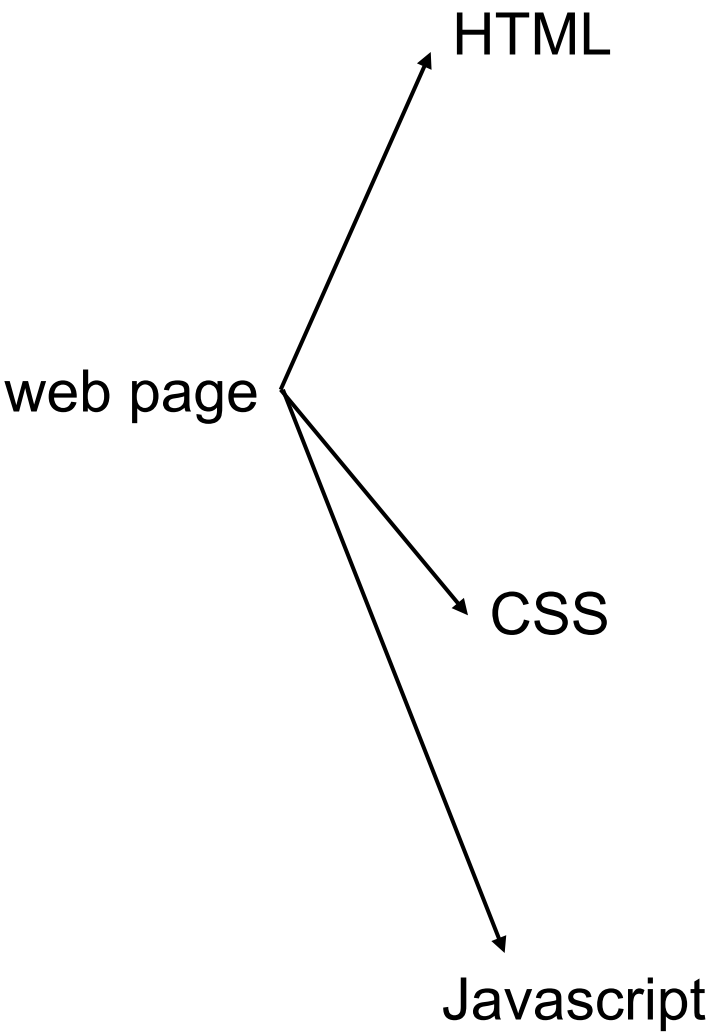
```
GET /account.html HTTP/1.1
Host: www.safebank.com
```

HTTP RESPONSE:

```
HTTP/1.0 200 OK
<HTML> . . . </HTML>
```

HTTP Response

Web page



HTML

A language to create structured documents
One can embed images, objects, or create interactive forms

index.html

```
<html>
    <body>
        <div>
            foo
            <a href="http://google.com">Go to Google!</a>
        </div>
        <form>
            <input type="text" />
            <input type="radio" />
            <input type="checkbox" />
        </form>
    </body>
</html>
```

CSS (Cascading Style Sheets)

Style sheet language used for describing the presentation of a document

index.css

```
p.serif {
font-family: "Times New Roman", Times, serif;
}
p.sansserif {
font-family: Arial, Helvetica, sans-serif;
}
```

Javascript

Programming language used to manipulate web pages. It is a high-level, untyped and interpreted language with support for objects.

Supported by all web browsers

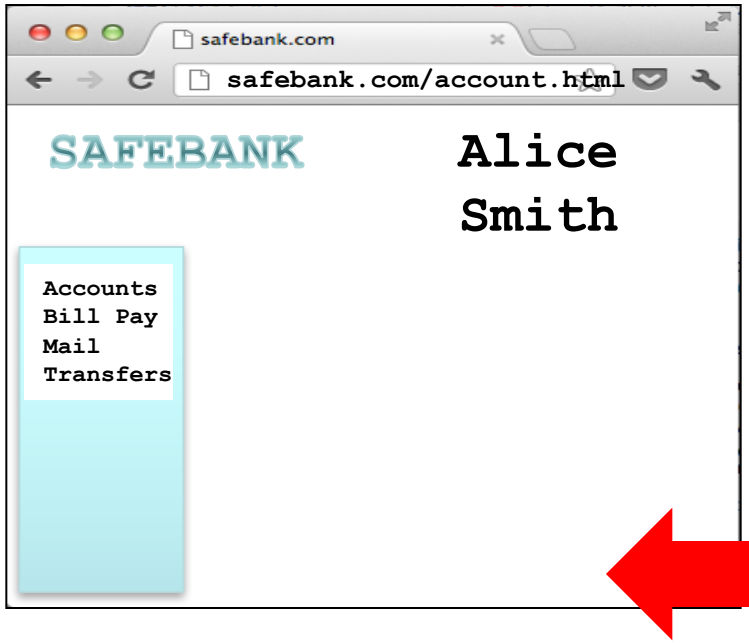
```
<script>
function myFunction() {
document.getElementById("demo").innerHTML = ”Text changed.";
}
</script>
```

Very powerful!

HTTP

CLIENT BROWSER

WEB SERVER



HTTP REQUEST:

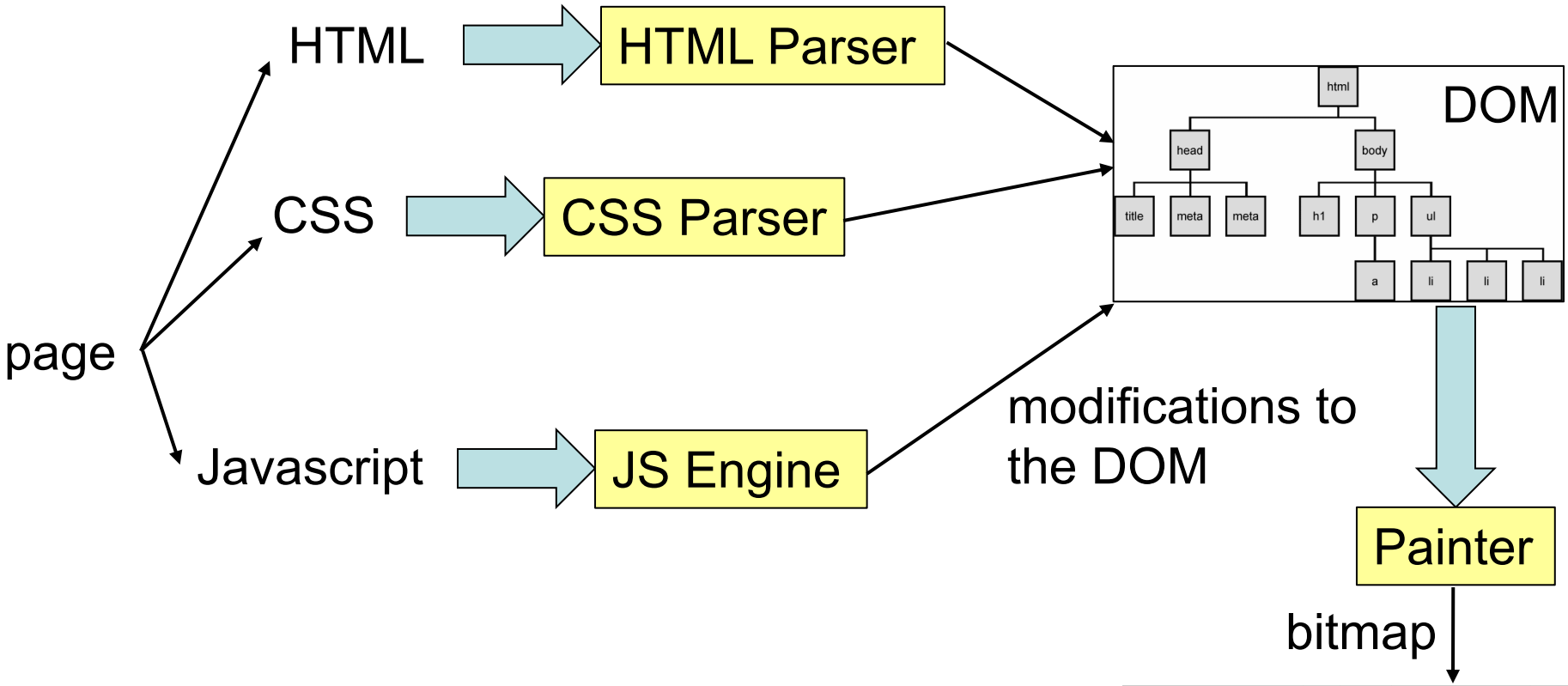
```
GET /account.html HTTP/1.1
Host: www.safebank.com
```

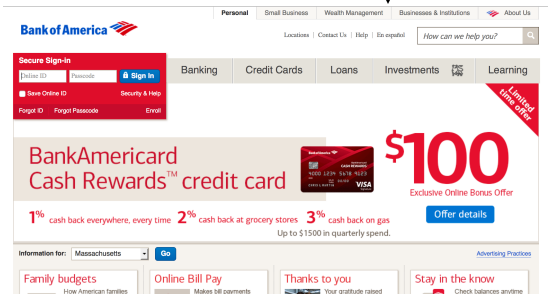
HTTP RESPONSE:

```
HTTP/1.0 200 OK
<HTML> . . . </HTML>
```

webpage

Page rendering



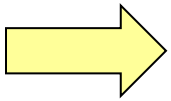


DOM (Document Object Model)

a cross-platform model for representing and interacting with objects in HTML

HTML

```
<html>
    <body>
        <div>
            foo
        </div>
        <form>
            <input type="text" />
            <input type="radio" />
            <input type="checkbox" />
        </form>
    </body>
</html>
```



DOM Tree

```
|-> Document
   |-> Element (<html>)
      |-> Element (<body>)
         |-> Element (<div>)
            |-> text node
         |-> Form
               |-> Text-box
               |-> Radio Button
               |-> Check Box
```

Javascript is very powerful

Almost anything you want to the DOM!

A JS script embedded on a page can modify in almost arbitrary ways the DOM of the page. The same happens if an attacker manages to get you load a script into your page.

w3schools.com has nice interactive tutorials

Example of what Javascript can do…

Can change HTML content:

```
<p id="demo">JavaScript can change HTML content.</p>

<button type="button"
onclick="document.getElementById('demo').innerHTML =
'Hello JavaScript!'">
   Click Me!</button>
```

DEMO from w3schools.com

Other examples

Can change images
Can change style of elements
Can hide elements
Can unhide elements
Can change cursor

Other example: can access cookies

Read cookie with JS:

```
var x = document.cookie;
```

Change cookie with JS:

```
document.cookie = "username=John Smith; expires=Thu,
18 Dec 2013 12:00:00 UTC; path=/";
```

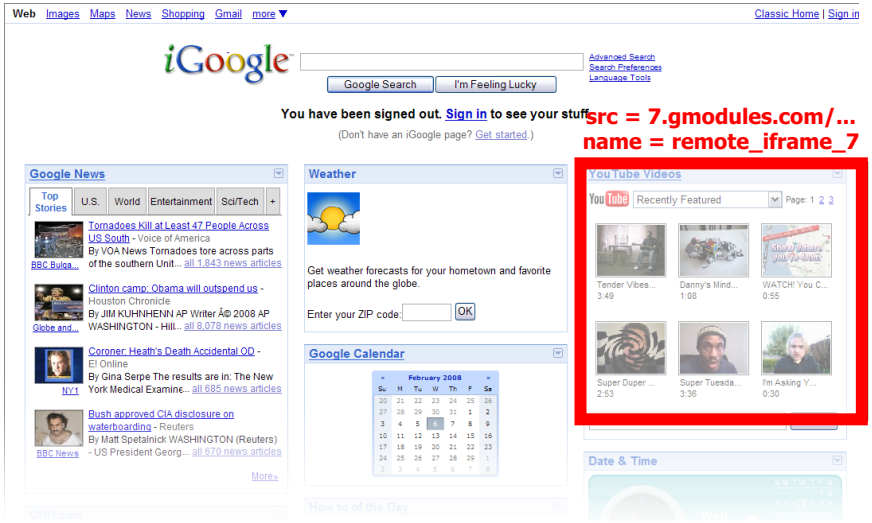
Frames

Frames

- Enable embedding a page within a page

```
<iframe src="URL"></iframe>
```

Frames



- Modularity
  - Brings together content from multiple sources
  - Client-side aggregation
- Delegation
  - Frame can draw only on its own rectangle

Slide from Dan Boneh

Frames

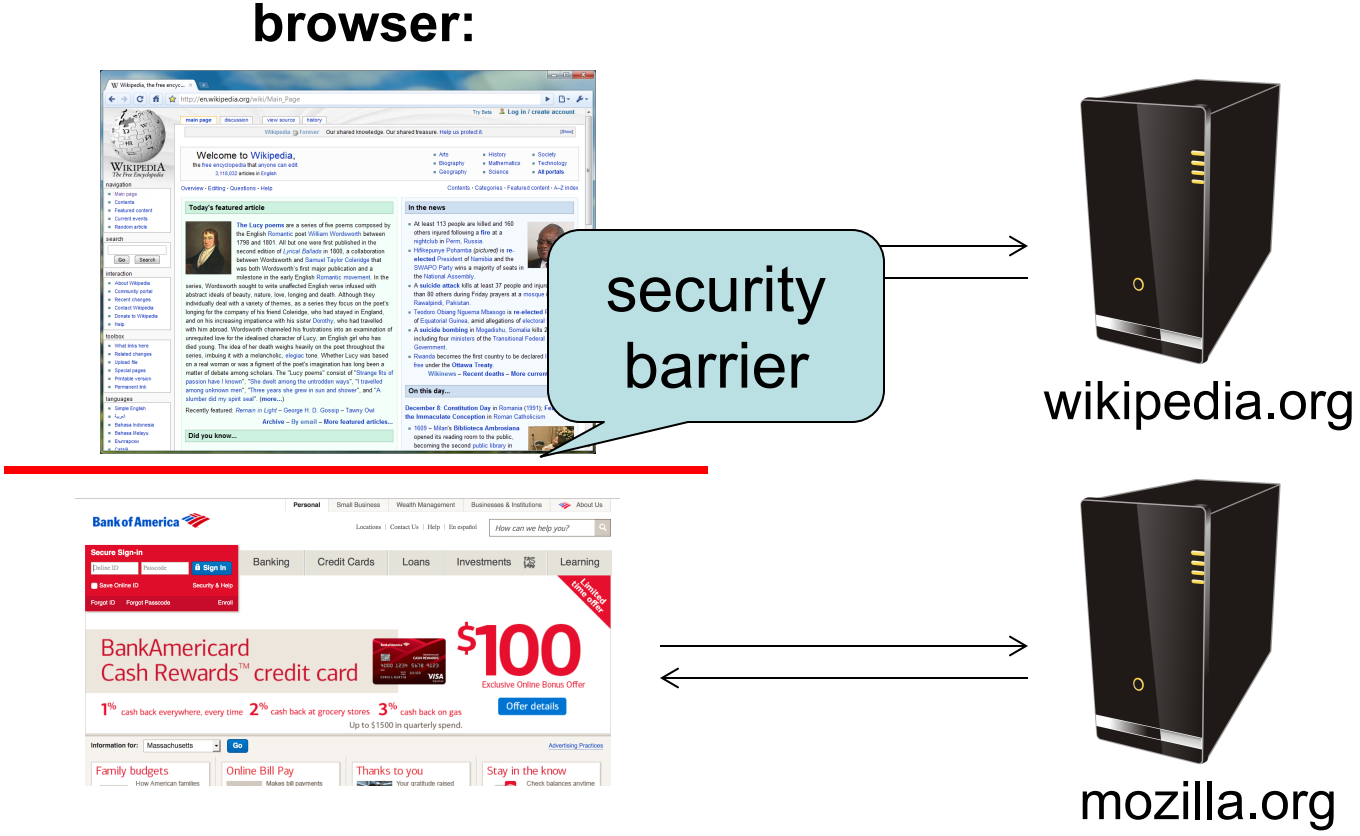
- Outer page can specify only sizing and placement of the frame in the outer page
- Frame isolation: Our page cannot change contents of inner page, inner page cannot change contents of outer page

Web security

Same-origin policy

Same-origin policy

- Each site in the browser is isolated from all others



Same-origin policy

- Multiple pages from the same site are not isolated

browser:

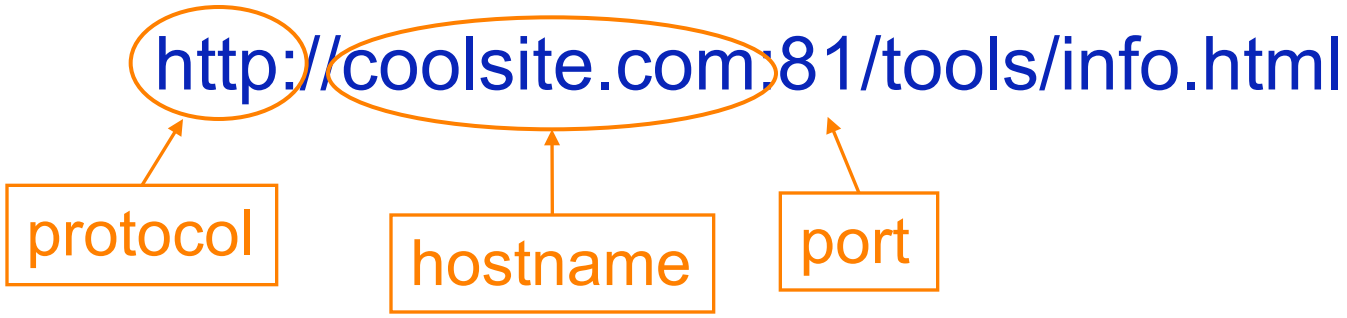
No security barrier

wikipedia.org

wikipedia.org

Origin

- Granularity of protection for same origin policy
- Origin = protocol + hostname + port



- It is **string matching**! If these match, it is same origin, else it is not. Even though in some cases, it is logically the same origin, if there is no match, it is not

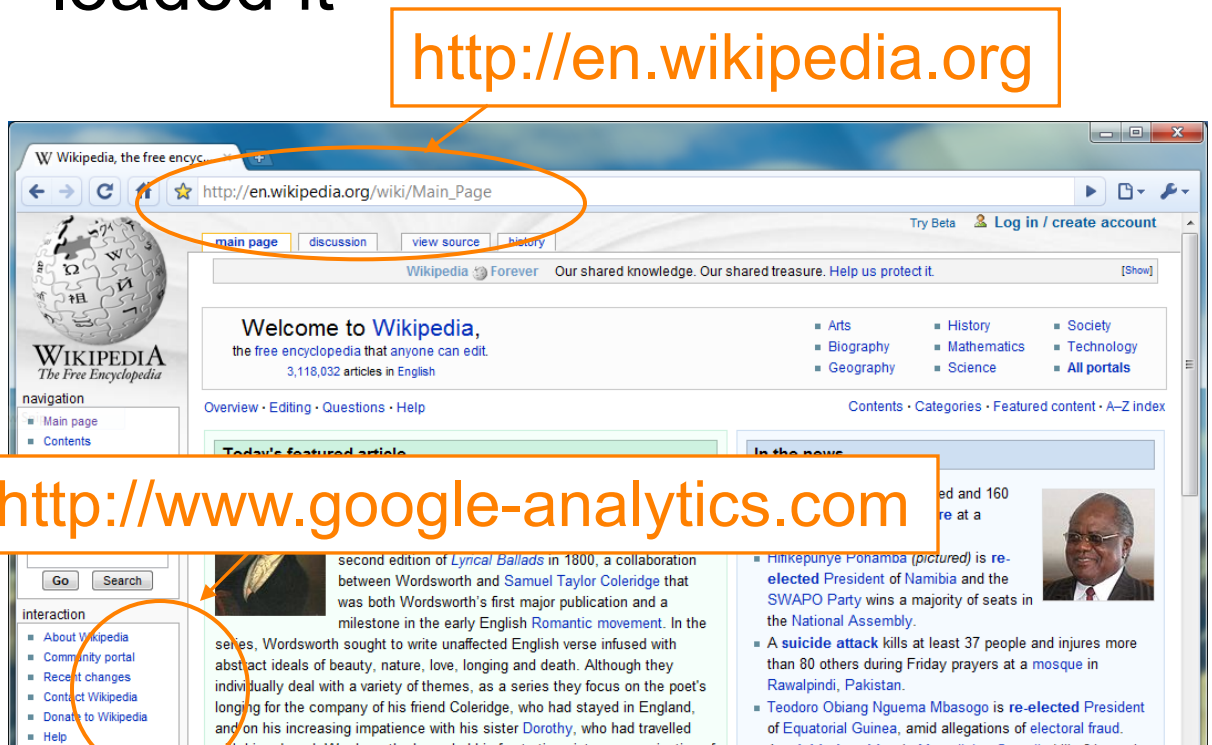
Same-origin policy

One origin should not be able to access the resources of another origin

Javascript on one page cannot read or modify pages from different origins

Same-origin policy

- The origin of a page is derived from the URL it was loaded from
- Javascript runs with the origin of the page that loaded it

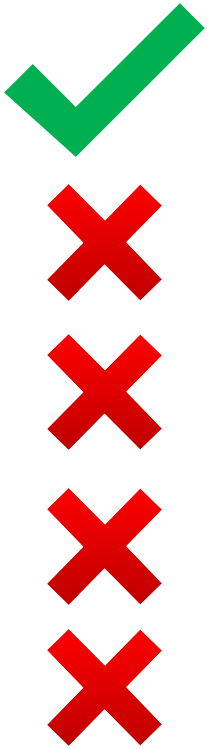


Origins of other components

- **<img src=“”>** the image is “copied” from the remote server into the new page so it has the origin of the embedding page (like JS) and not of the remote origin
- iframe: origin of the URL from which the iframe is served, and not the loading website.

Exercises

Originating document	Accessed document
http://wikipedia.org/**a**/	http://wikipedia.org/**b**/
http://wikipedia.org/	http://**www.**wikipedia.org/
http://wikipedia.org/	**https**://wikipedia.org/
http://wikipedia.org**:81**/	http://wikipedia.org**:82**/
http://wikipedia.org**:81**/	http://wikipedia.org/

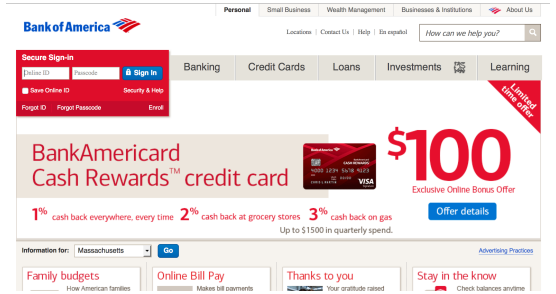


except  !!!

Cross-origin communication

- Allowed through a narrow API: **postMessage**
- Receiving origin decides if to accept the message based on origin (whose correctness is enforced by browser)

```
postMessage
("run this
script",
script)
```



Check origin, and request!

Chromodo
Private Internet Browser

Fast and versatile Internet Browser based on Chromium, with highest levels of speed, security and privacy!

Issue 704: Comodo: Comodo "Chromodo" Browser disables same origin policy, Effectively turning off web security.

13 people starred this issue and may be notified of changes.

tus: Fixed
ner: tav...@google.com
sed: Yesterday
project-...@google.com
dor-Comodo
duct-Chromodo
rity-critical

Project Member Reported by tav...@google.com, Jan 21, 2016

When you install Comodo Internet Security, by default a new browser called Chromodo is installed and set as the default browser. Additionally, all shortcuts are replaced with Chromodo links and all settings, cookies, etc are imported from Chrome. They also hijack DNS settings, among other shady practices.

https://www.comodo.com/home/browsers-toolbars/chromodo-private-internet-browser.php

Chromodo is described as "highest levels of speed, security and privacy", but actually disables all web security. Let me repeat that, they ***disable the same origin policy***.... ?!?..

HTTP cookies

Outrageous Chocolate Chip Cookies

1676 reviews

Made 321 times

Recipe by: Joan

"A great combination of chocolate chips, oatmeal, and peanut butter."

Save | I Made it | Rate it | Share | Print

Ingredients

25 m | 18 servings | 207 cals

- 1/2 cup butter
- 1/2 cup white sugar
- 1/3 cup packed brown sugar
- 1 cup all-purpose flour
- 1 teaspoon baking soda
- 1/4 teaspoon salt
- 1/2 cup rolled oats
- 1 cup semisweet chocolate chips

On Sale

On

What's on sale near you.

TARGET

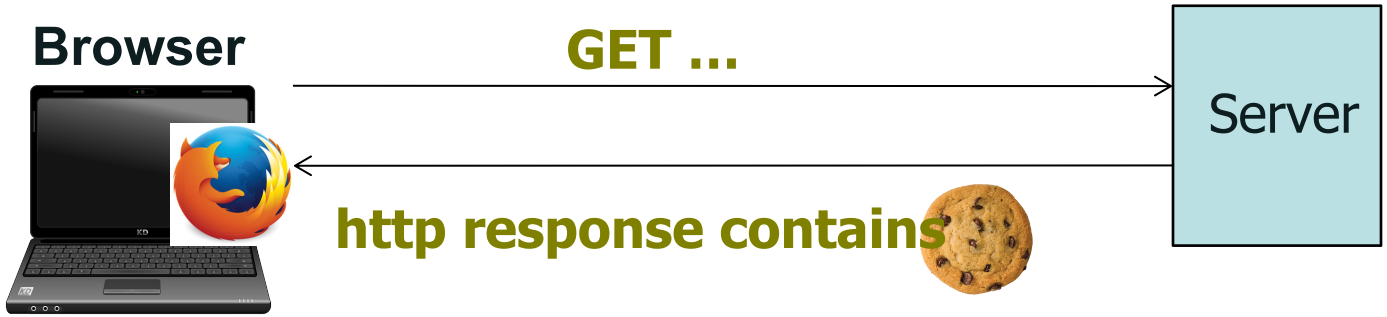
Target
1057 Eastshore Hwy
ALBANY, CA 94710
Sponsored

May we suggest

These nearby stores have ingredients on sale!

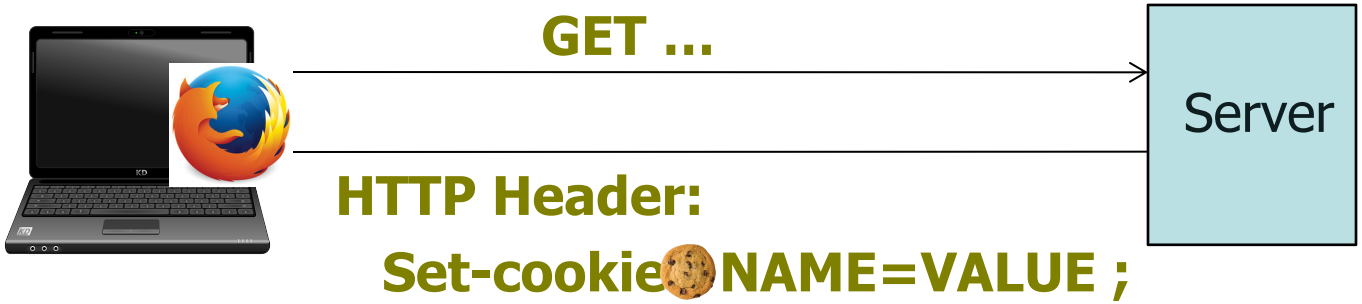
Cookies

- A way of maintaining state



`Browser maintains cookie jar`

Setting/deleting cookies by server



- The first time a browser connects to a particular web server, it has no cookies for that web server
- When the web server responds, it includes a **Set-Cookie:** header that defines a cookie
- Each cookie is just a name-value pair

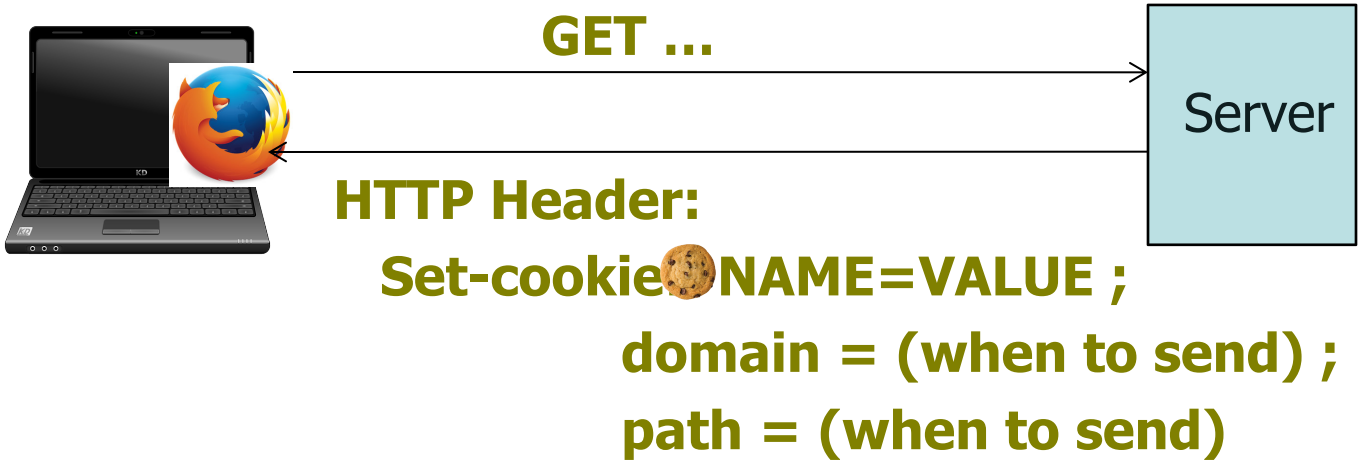
View a cookie

In a web console (firefox, tool->web developer->web console), type

```
document.cookie
```

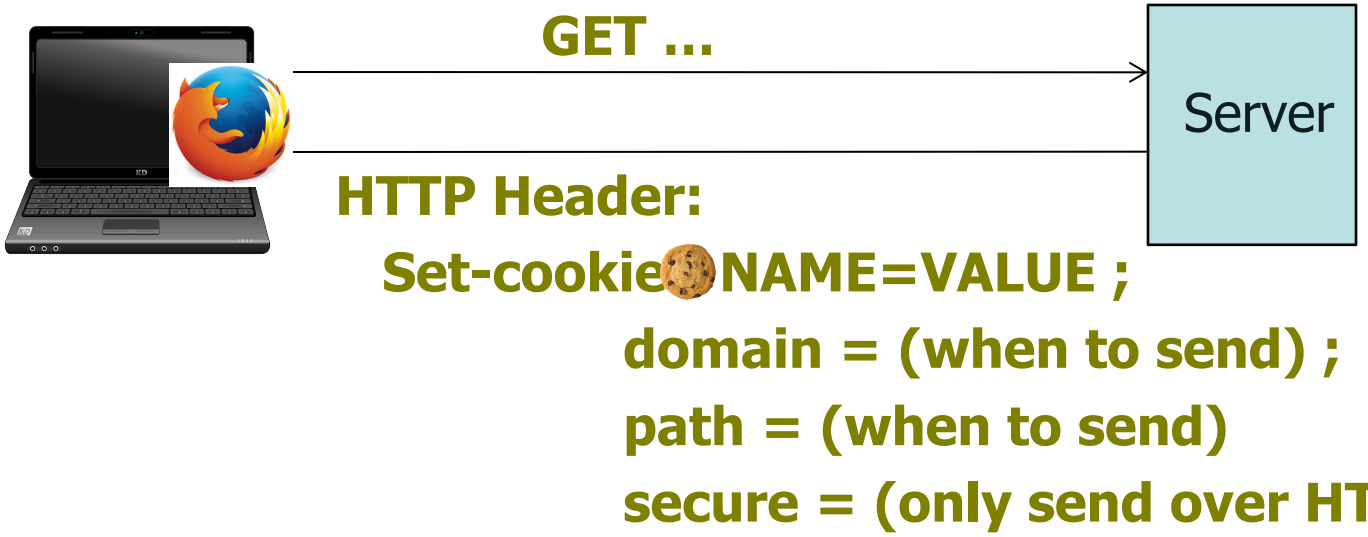
to see the cookie for that site

Cookie scope



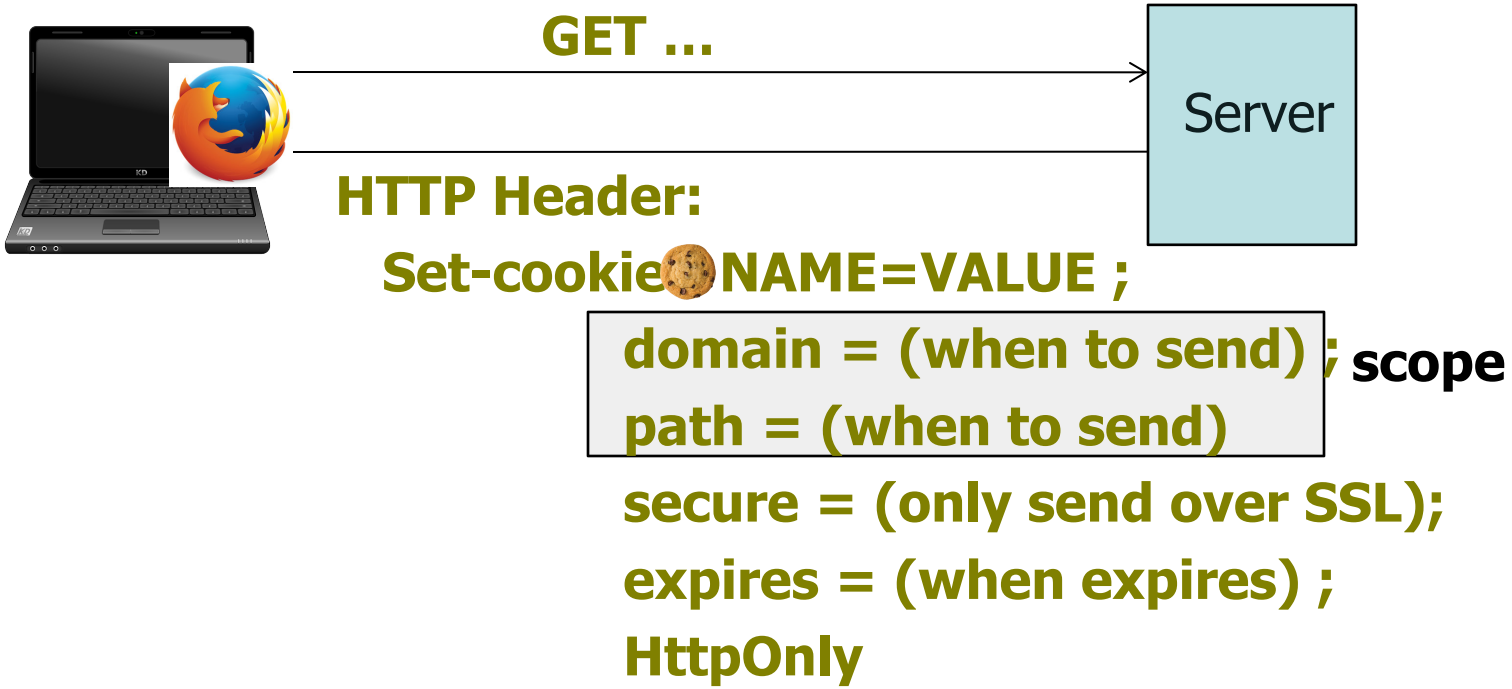
- When the browser connects to the same server later, it includes a Cookie: header containing the name and value, which the server can use to connect related requests.
- Domain and path inform the browser about which sites to send this cookie to

Cookie scope



- **Secure: sent over https only**
  - **https provides secure communication (privacy and integrity)**

Cookie scope



- Expires is expiration date
  - Delete cookie by setting “expires” to date in past
- HttpOnly: cookie cannot be accessed by Javascript, but only sent by browser

Cookie scope

- Scope of cookie might not be the same as the URL-host name of the web server setting it
  - **Different from same-origin policy!!**

Rules on:

1. What scopes a URL-host name is allowed to set
2. When a cookie is sent to a host

What scope a server may set for a cookie

The browser checks if the server may set the cookie, and if not, it will not accept the cookie.

domain: any domain-suffix of URL-hostname, except TLD

example: host = "login.site.com" [top-level domains, e.g. '.com']

allowed domains	**disallowed domains**
login.site.com	**user.site.com**
.site.com	**othersite.com**
	.com

⇒ **login.site.com** can set cookies for all of **.site.com**
but not for another site or TLD

path: can be set to anything

Examples

Web server at foo.example.com wants to set cookie with domain:

domain	Whether it will be set, and if so, where it will be sent to
(value omitted)	*foo.example.com* (exact)
bar.foo.example.com	
foo.example.com	**.foo.example.com*
baz.example.com	
example.com	
ample.com	
.com	

Credits: The Tangled Web: A Guide to Securing Modern Web Applications, by Michał Zalewski

Examples

Web server at foo.example.com wants to set cookie with domain:

domain	Whether it will be set, and if so, where it will be sent to
(value omitted)	*foo.example.com* (exact)
bar.foo.example.com	Cookie not set: domain more specific than origin
foo.example.com	**.foo.example.com*
baz.example.com	Cookie not set: domain mismatch
example.com	**.example.com*
ample.com	Cookie not set: domain mismatch
.com	Cookie not set: domain too broad, security risk

Credits: The Tangled Web: A Guide to Securing Modern Web Applications, by Michał Zalewski

When browser sends cookie

Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain, and
- cookie-path is prefix of URL-path, and
- [protocol=HTTPS if cookie is “secure”]

When browser sends cookie

A cookie with

domain = example.com, and

path = /some/path/

will be included on a request to

http://foo.example.com/some/path/subdirectory/hello.txt

Examples: Which cookie will be sent?

cookie 1
name = userid
value = u1
domain = login.site.com
path = /
non-secure

cookie 2
name = userid
value = u2
domain = .site.com
path = /
non-secure

http://checkout.site.com/	**cookie: userid=u2**
http://login.site.com/	**cookie: userid=u1, userid=u2**
http://othersite.com/	**cookie: none**

Examples

cookie 1
name = userid
value = u1
domain = login.site.com
path = /
secure

cookie 2
name = userid
value = u2
domain = .site.com
path = /
non-secure

http://checkout.site.com/ **cookie: userid=u2**

http://login.site.com/ **cookie: userid=u2**

https://login.site.com/ **cookie: userid=u1; userid=u2**

(arbitrary order)

Client side read/write: document.cookie

- Setting a cookie in Javascript:

  document.cookie = “name=value; expires=…; ”

- Reading a cookie:
  - alert(document.cookie): prints string containing all cookies available for document (based on [protocol], domain, path)
- Deleting a cookie:

  document.cookie = “name=; expires= Thu, 01-Jan-70”

document.cookie often used to customize page in Javascript

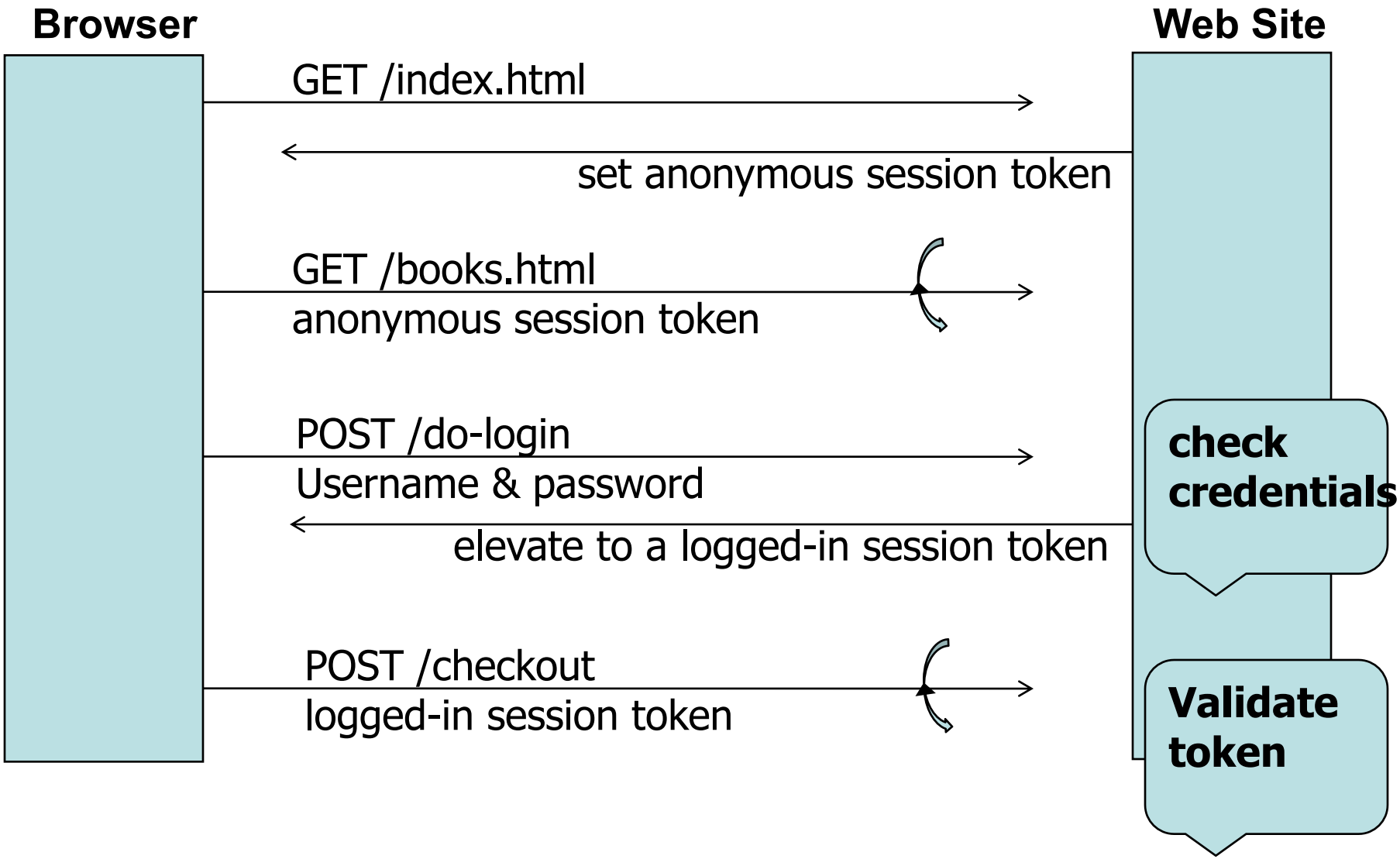
Sessions

Sessions

- A sequence of requests and responses from one browser to one (or more) sites
  - Session can be long (Gmail - two weeks)
    or short (e.g., banking)
  - without session mgmt:

    users would have to constantly re-authenticate
- Session management:
  - Authorize user once;
  - All subsequent requests are tied to user

Session tokens



Storing session tokens:
Lots of options (but none are perfect)

- Browser cookie:

  Set-Cookie: SessionToken=fduhye63sfdb

- Embed in all URL links:

  https://site.com/checkout?SessionToken=kh7y3b

- In a hidden form field:

  <input type="hidden" name="sessionid"
  value="kh7y3b">

Can you see problems with these?

Storing session tokens: problems

- Browser cookie:

  browser sends cookie with every request, even when it should not (see CSRF attack)

- Embed in all URL links:

  token leaks via HTTP Referer header (your browser tells a site which previous site it visited last in the Referer header, which may contain session tokens)

- In a hidden form field: short sessions only

Best answer: a combination of all of the above.

Questions?

- Same-origin policy
- DOM model
- Cookie policy
- Sessions