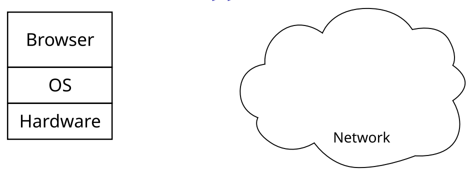
CSci 4271W
Development of Secure Software Systems
Day 20: Web security part 1: intro and privacy

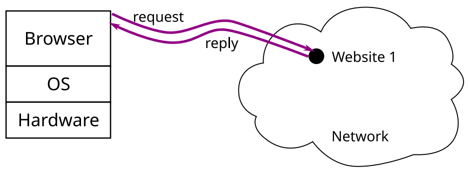
Stephen McCamant (he/him)
University of Minnesota, Computer Science & Engineering

Web applications



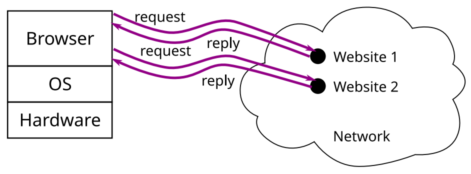
- The browser sends a request to a server
- The server processes this request, and sends a reply
- The browser receives data and code
- This may result in the need to send additional requests

Web applications



- The browser sends a request to a server
- The server processes this request, and sends a reply
- The browser receives data and code
- This may result in the need to send additional requests

Web applications



- The browser sends a request to a server
- The server processes this request, and sends a reply
- The browser receives data and code
- This may result in the need to send additional requests

Hypertext Transport Protocol

- HTTP is a stateless request/response protocol

Hypertext Transport Protocol

- HTTP is a stateless request/response protocol
- Clients send resource requests (usually GET, POST or PUT)

Hypertext Transport Protocol

- HTTP is a stateless request/response protocol
- Clients send resource requests (usually GET, POST or PUT)
- Servers send responses (info/success/redirect/error)

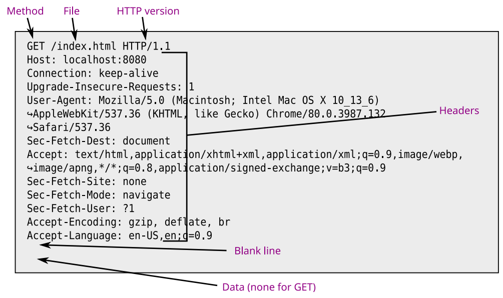
Hypertext Transport Protocol

- HTTP is a stateless request/response protocol
- Clients send resource requests (usually GET, POST or PUT)
- Servers send responses (info/success/redirect/error)
- Response bodies can reference additional resources

Hypertext Transport Protocol

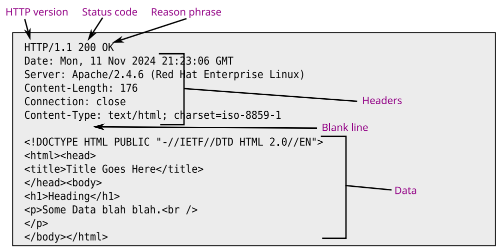
- HTTP is a stateless request/response protocol
- Clients send resource requests (usually GET, POST or PUT)
- Servers send responses (info/success/redirect/error)
- Response bodies can reference additional resources
- Most applications build stateful sessions on top of HTTP

HTTP GET request example



```
GET /index.html HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
↪AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132
↪Safari/537.36
Sec-Fetch-Dest: document
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,
↪image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

```

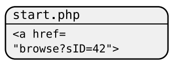
HTTP GET response example



```
HTTP/1.1 200 OK
Date: Mon, 11 Nov 2024 21:23:06 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux)
Content-Length: 176
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>Title Goes Here</title>
</head><body>
<h1>Heading</h1>
<p>Some Data blah blah.<br />
</p>
</body></html>
```

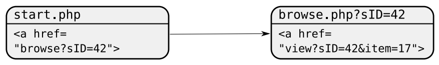
One kind of session

Using GET method:



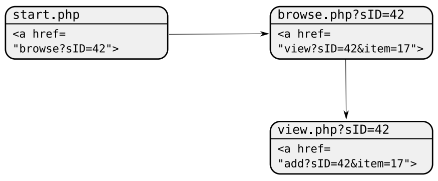
One kind of session

Using GET method:



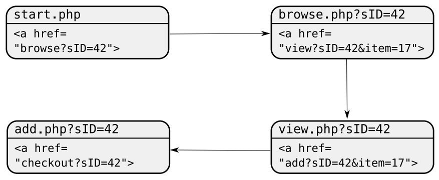
One kind of session

Using GET method:



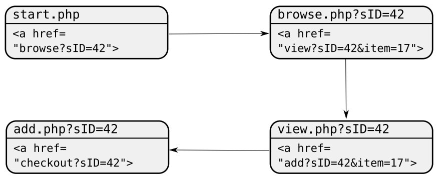
One kind of session

Using GET method:



One kind of session

Using GET method:



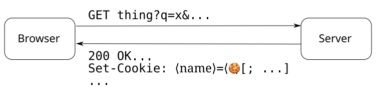
Application on server tracks changes to session

Cookies

Are the most prominent example of local storage

Cookies

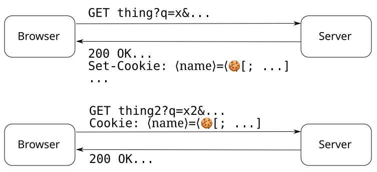
Are the most prominent example of local storage



```
GET thing?q=x&...

200 OK...
Set-Cookie: ⟨name⟩=⟨🍪[; ...]
...
```

Cookies

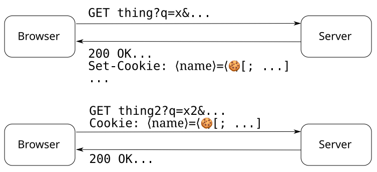
Are the most prominent example of local storage



```
GET thing?q=x&...

200 OK...
Set-Cookie: ⟨name⟩=⟨🍪[; ...]
...

GET thing2?q=x2&...
Cookie: ⟨name⟩=⟨🍪[; ...]

200 OK...
```

Cookies

Are the most prominent example of local storage



```
GET thing?q=x&...

200 OK...
Set-Cookie: ⟨name⟩=⟨🍪[; ...]
...

GET thing2?q=x2&...
Cookie: ⟨name⟩=⟨🍪[; ...]

200 OK...
```

Local storage allows web applications to store some session state with the client.

Embedded content

HTML documents can reference many other resources:

- Style sheets – influence display of elements
- Scripts – `<script src="nextslide.js" />`
- Frames – include other pages
- Images – loaded and displayed with separate requests

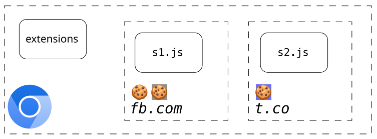
Scripts

JavaScript embedded in a page runs in a sandbox but can:

- Manipulate page's Document Object Model (DOM), adding or removing elements
- Make additional HTTP requests
- Open windows, capture user input
- Access page's local storage
- Interact with browser API

Security goals

Like OSes, browsers provide uniform resource access and attempt to protect applications from each other.
The unit of protection is the "origin" (informally, "domain")



Data associated with a page originating from domain A should not be leaked to or altered by a page originating from domain B. (The "same-origin policy".)

Example: `https://z.umn.edu/twostop`

Select "Network" tab in Chrome Developer Tools, then click on CSCI current term.

- What is the IP address of the server?
- What webserver application is running on the server?
- What cookies are set?
- How many script objects are included? CSS?
- What line is the table of classes on?
- What HTTP method does the subject/term form use?

Outline

Web basics and security model

Announcements intermission

Web privacy vs. tracking

Assignments, other logistics

- Project 2 section drafts are due tonight
- Project 2 reports are due next Tuesday
  - Gradescope and Canvas entries for both exist now
- Project 1 feedback coming ASAP
- Final exam location confirmed: same as lectures and midterms
  - Saturday May 10th, 4–6pm, 3-115 Keller Hall

Outline

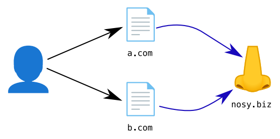
Web basics and security model

Announcements intermission

Web privacy vs. tracking

Tracking

One threat to users is tracking: "data brokers" collect user "profiles" from pages visited, location, etc.
This info is then used to target ads, extract more sales, sold to other companies, etc.



Web bugs

One tracking mechanism is the "web bug": `a.com` pages cause the browser to send a request to `nosy.biz`

```
<img src=... />
<iframe src=... />
```

Also common in a 1-pixel by 1-pixel size.

CSS and history

```
<style type="text/css">
body {font-family: sans-serif;}
a.test1:visited {background-image:url('test1.png');}
a.test2:visited {background-image:url('test2.png');}
</style>
<body>
<a href="test1.html" class="test1" />
<a href="test2.html" class="test2" />
</body>
```

Cookies for tracking

Setting unique cookies per browser allows servers to:

- Track clients across networks
- Record location history
- Track across web sites (helped by referrer headers)

Cache cookies (1)

A tracking mechanism based on storing data in the browser's cache.

```
<iframe src="cookify-me.php" width=0 height=0 />
```

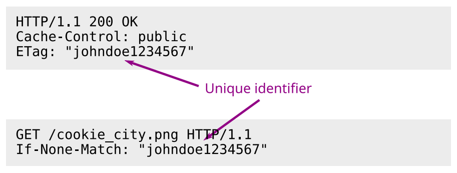
```
HTTP/1.1 200 OK
Date: Wed, 13 Apr 2024 17:32:25 GMT
Expires: 19 Jan 2038 03:14:06 GMT
Cache-Control: public

<html><body><img src="1234567.png" /></body></html>
```

Reloaded every visit

Cache cookies (2)

A tracking mechanism based on storing data in the browser's cache.

```
<img src="cookie_city.png" width=0 height=0 />
```



```
HTTP/1.1 200 OK
Cache-Control: public
ETag: "johndoe1234567"
```

```
GET /cookie_city.png HTTP/1.1
If-None-Match: "johndoe1234567"
```

Example: `https://www.cnn.com`

Select "Network" tab in Chrome Developer Tools, then pick a story.

- What request headers are set?
- What security-relevant response headers are set?
- How many script objects are included? CSS?

Countermeasures

`DNT: 1` (from "do not track") was a proposed voluntary anti-tracking HTTP header. There was never sufficient agreement for it to be effective.

Extensions: AdBlock+, NoScript, RequestPolicy, Ghostery, PrivacyBadger

"Private Browsing" and "Guest" modes isolate browser state. Guest mode usually provides more isolation.