Web Security Model

Lecture 10

Software Security Engineering

Winter 2023
Thompson Rivers University

Web Security

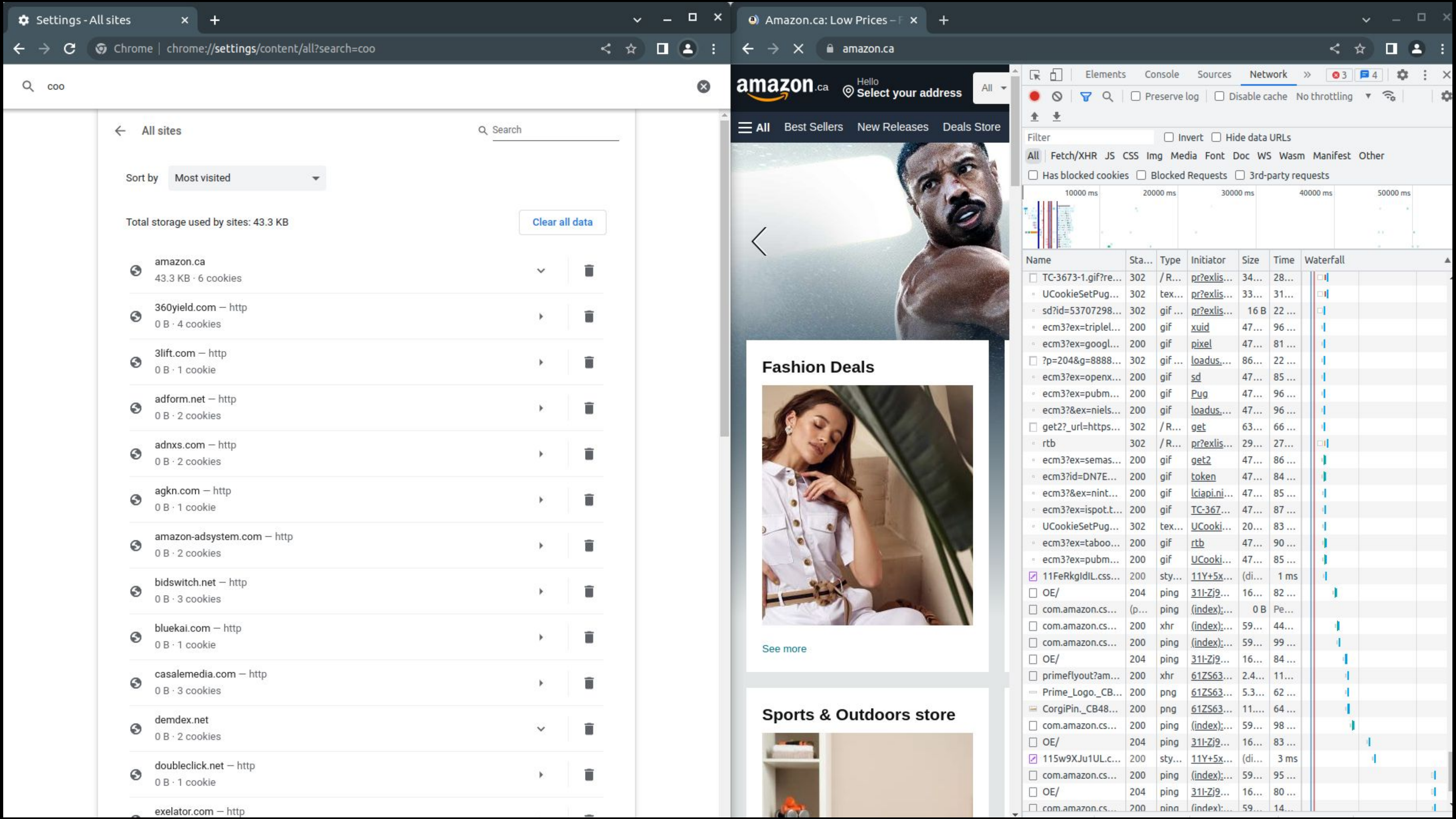
- making websites safe to visit
- protecting user data across websites
- allowing bad sites to run without interacting with good sites
- supporting secure web applications
- securing internet traffic (e.g., TLS)

Web Security is not as simple as it seems

Web Security is not as simple as it seems

because of the various third-party websites and resources
that are loaded when visiting a website

let’s open Amazon.ca



All sites

Sort by Most visited

Total storage used by sites: 43.3 KB

Clear all data

amazon.ca
43.3 KB · 6 cookies

360yield.com — http
0 B · 4 cookies

3lift.com — http
0 B · 1 cookie

adform.net — http
0 B · 2 cookies

adnxs.com — http
0 B · 2 cookies

agkn.com — http
0 B · 1 cookie

amazon-adsystem.com — http
0 B · 2 cookies

bidswitch.net — http
0 B · 3 cookies

bluekai.com — http
0 B · 1 cookie

casalemedia.com — http
0 B · 3 cookies

demdex.net
0 B · 2 cookies

doubleclick.net — http
0 B · 1 cookie

exelator.com — http

Name	Sta...	Type	Initiator	Size	Time
TC-3673-1.gif?re...	302	/ R...	pr?exlis...	34...	28...
UCookieSetPug...	302	tex...	pr?exlis...	33...	31...
sd?id=53707298...	302	gif ...	pr?exlis...	16 B	22 ...
ecm3?ex=triplel...	200	gif	xuid	47...	96 ...
ecm3?ex=googl...	200	gif	pixel	47...	81 ...
?p=204&g=8888...	302	gif ...	loadus....	86...	22 ...
ecm3?ex=openx...	200	gif	sd	47...	85 ...
ecm3?ex=pubm...	200	gif	Pug	47...	96 ...
ecm3?&ex=niels...	200	gif	loadus....	47...	96 ...
get2?_url=https...	302	/ R...	get	63...	66 ...
rtb	302	/ R...	pr?exlis...	29...	27...
ecm3?ex=semas...	200	gif	get2	47...	86 ...
ecm3?id=DN7E...	200	gif	token	47...	84 ...
ecm3?&ex=nint...	200	gif	lciapi.ni...	47...	85 ...
ecm3?ex=ispot.t...	200	gif	TC-367...	47...	87 ...
UCookieSetPug...	302	tex...	UCooki...	20...	83 ...
ecm3?ex=taboo...	200	gif	rtb	47...	90 ...
ecm3?ex=pubm...	200	gif	UCooki...	47...	85 ...
11FeRkgldIL.css...	200	sty...	11Y+5x...	(di...	1 ms
OE/	204	ping	31I-Zj9...	16...	82 ...
com.amazon.cs...	(p...	ping	(index):...	0 B	Pe...
com.amazon.cs...	200	xhr	(index):...	59...	44...
com.amazon.cs...	200	ping	(index):...	59...	99 ...
OE/	204	ping	31I-Zj9...	16...	84 ...
primeflyout?am...	200	xhr	61ZS63...	2.4...	11...
Prime_Logo._CB...	200	png	61ZS63...	5.3...	62 ...
CorgiPin._CB48...	200	png	61ZS63...	11....	64 ...
com.amazon.cs...	200	ping	(index):...	59...	98 ...
OE/	204	ping	31I-Zj9...	16...	83 ...
115w9XJu1UL.c...	200	sty...	11Y+5x...	(di...	3 ms
com.amazon.cs...	200	ping	(index):...	59...	95 ...
OE/	204	ping	31I-Zj9...	16...	80 ...
com.amazon.cs...	200	ping	(index):...	59...	14...

Web Background

URI URN URL

- uniform resource indicator (URI)
  - a unique string meant to identify some resource
- uniform resource name (URN)
  - a kind of URI that gives a unique name
  - urn:isbn:0130460192 (link)
    - International Standard Book Number
  - urn:ietf:rfc:2648 (link)
    - Request for Comment
  - urn:lex:eu:council:directive:2010-03-09;2010-19-UE (link)
    - laws and other legal norms

URI URN URL (cont.)

- uniform resource locator (URL)
  - most common kind of URI
  - identifies the resource
  - also gives a sequence of instructions on how to get it
- format: schema:[//[user@]host[:port]]path[?query][#fragment]
  - [item] is optional
  - host include www.tru.ca
  - schema include http, ftp
  - ports omitted if default for the schema

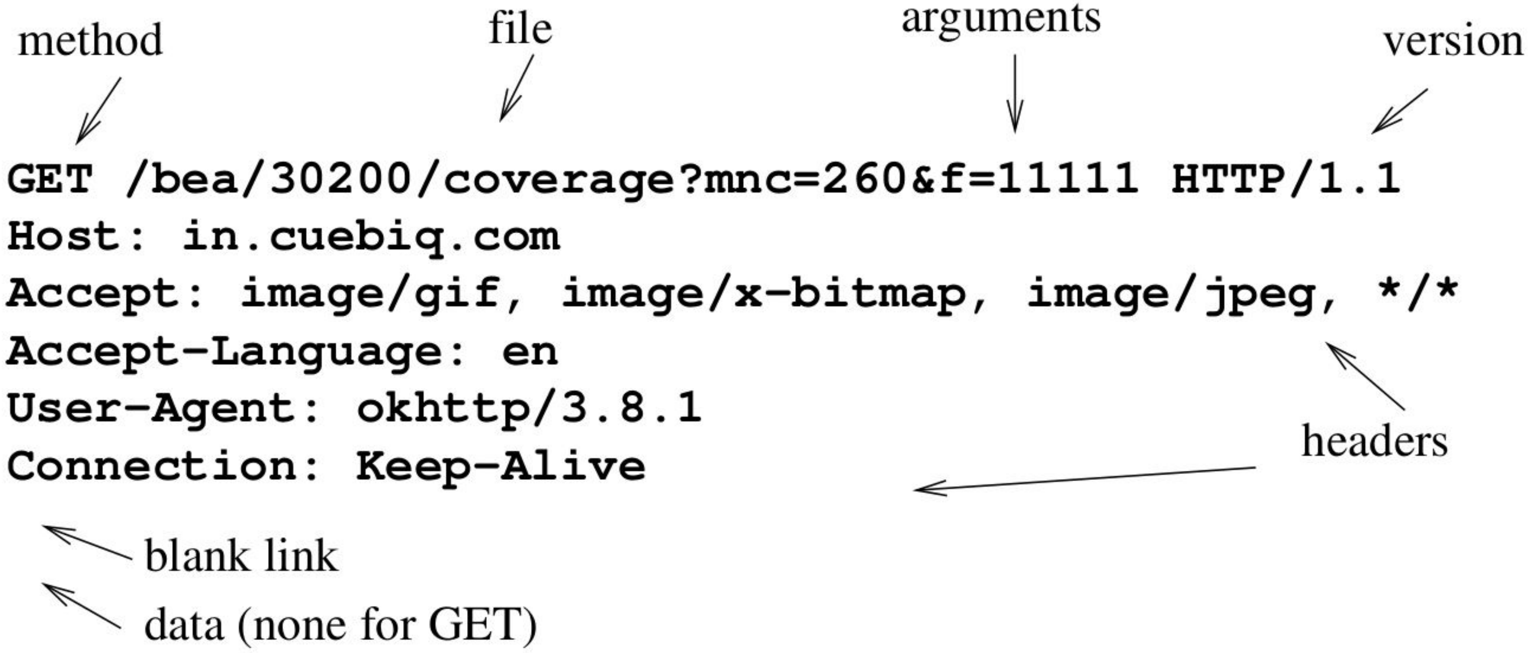
Host or Domain

- a series of one or more dot-separated parts
  - www.tru.ca
  - localhost
- top level domain (TLD): com, ca, org, net
- second and third level domains are subordinate to their parent
  - .ca is top of the tree, then tru, then the www server
  - you can add more myserver.eng.tru.ca
  - in practice, to add a subdomain, create a new DNS record on your domain's DNS hosting provider's website that maps the subdomain name to an IP address or hostname.

HTTP: HyperText Transfer Protocol

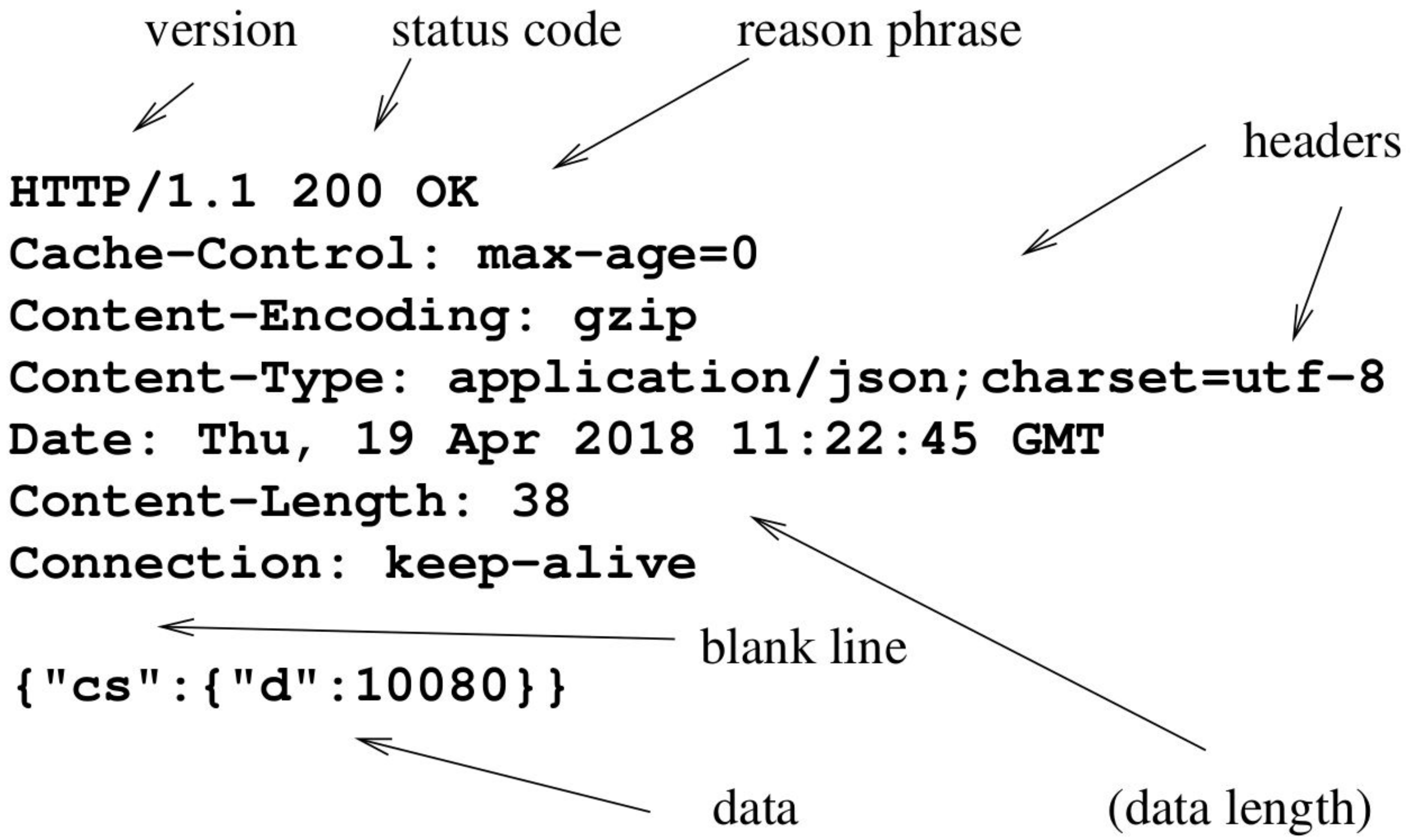
- used to request and return data
    - GET, POST, HEAD, …
- stateless request/response protocol
    - each request is independent of previous requests
    - statelessness has a significant impact on design and implementation of applications
    - stateless and keep-alive connections are two distinct concepts
    - we can use techniques such as cookies to maintain state

HTTP Request



```
GET /bea/30200/coverage?mnc=260&f=11111 HTTP/1.1
Host: in.cuebiq.com
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
User-Agent: okhttp/3.8.1
Connection: Keep-Alive
```

HTTP Response



```
HTTP/1.1 200 OK
Cache-Control: max-age=0
Content-Encoding: gzip
Content-Type: application/json;charset=utf-8
Date: Thu, 19 Apr 2018 11:22:45 GMT
Content-Length: 38
Connection: keep-alive

{"cs":{"d":10080}}
```

Common HTTP response status codes

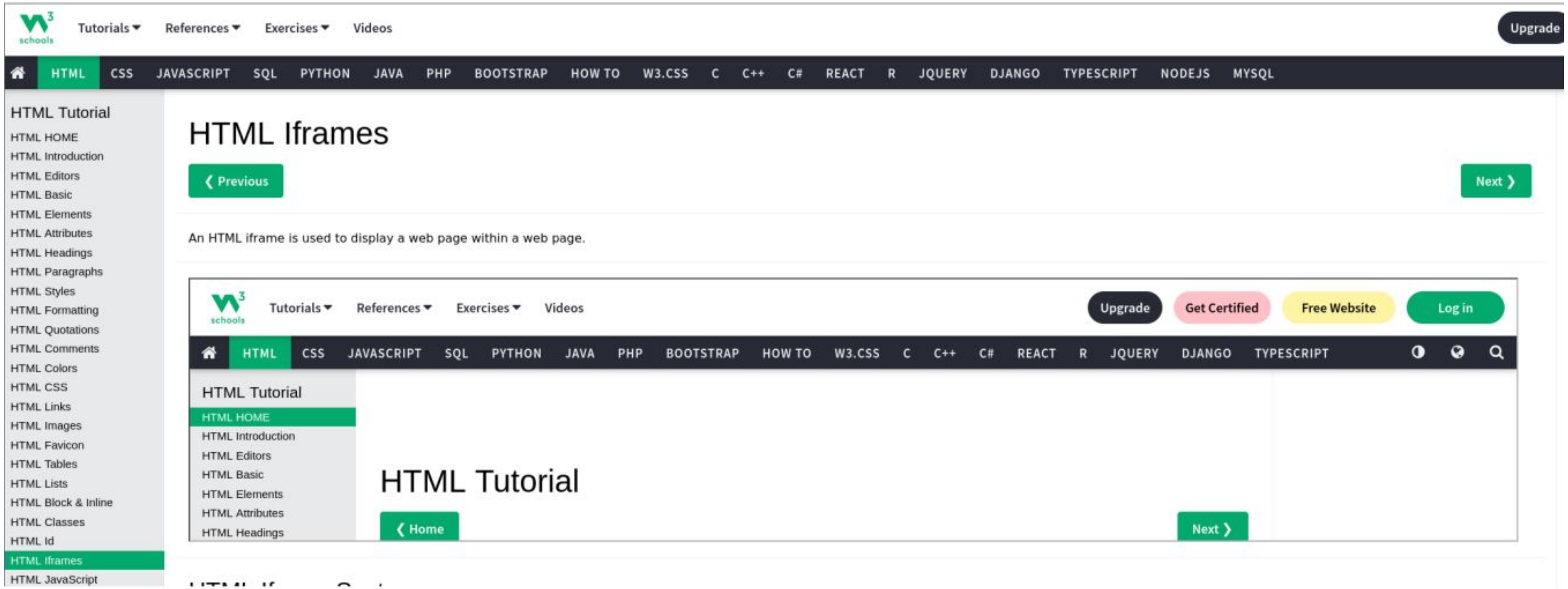
- 200 OK
- 201 Created (typically in response to a POST request or a PUT request)
- 204 No Content
- 301 Moved Permanently
- 302 Found
- 304 Not Modified
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 503 Service Unavailable

Goals of Web Security

- safely browse the Web
  - malicious website cannot:
    - steal information from the user
    - modify legitimate sites
    - otherwise harm the user
- support secure web applications
  - applications delivered over the web should have the same security as local software

Goals of Web Security (cont.)

- security should be provided even if the user visits both a good and bad site
- evil sites should not be able to interfere with good sites
- even when they are run
    - at the same time
    - separate window
    - separate tab
    - in an iframe on the same webpage
        - i.e., evil site presents good site inside



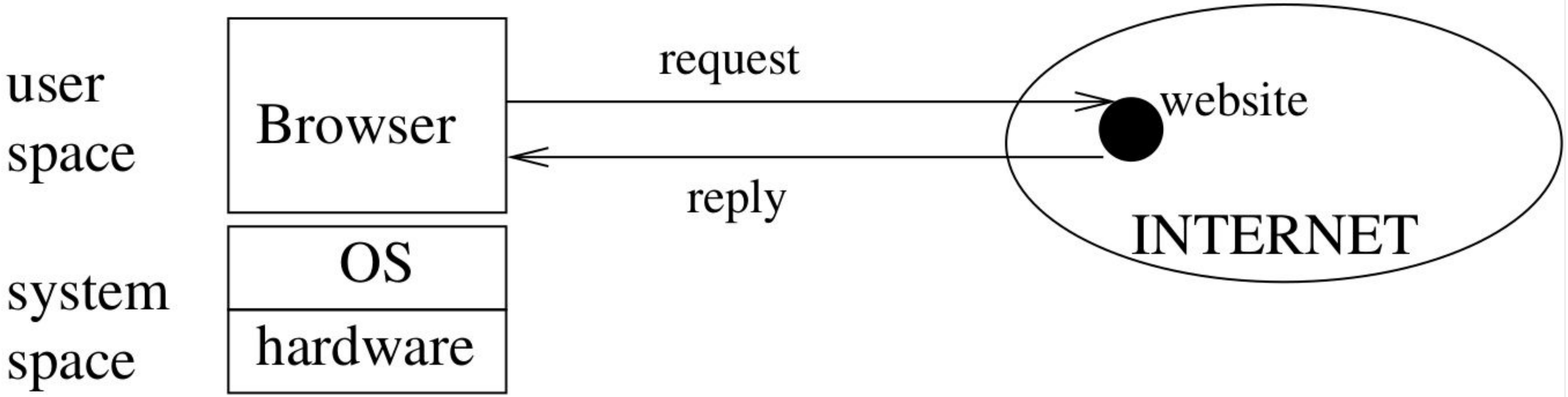
HTML Iframes

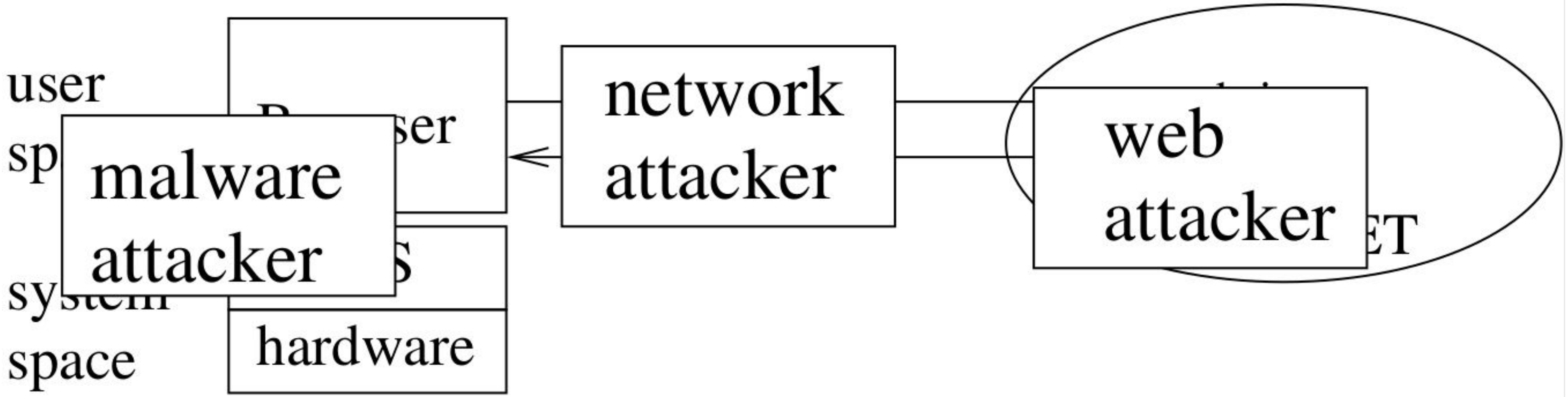
An HTML iframe is used to display a web page within a web page.

HTML Tutorial

Two Sides of Web Security

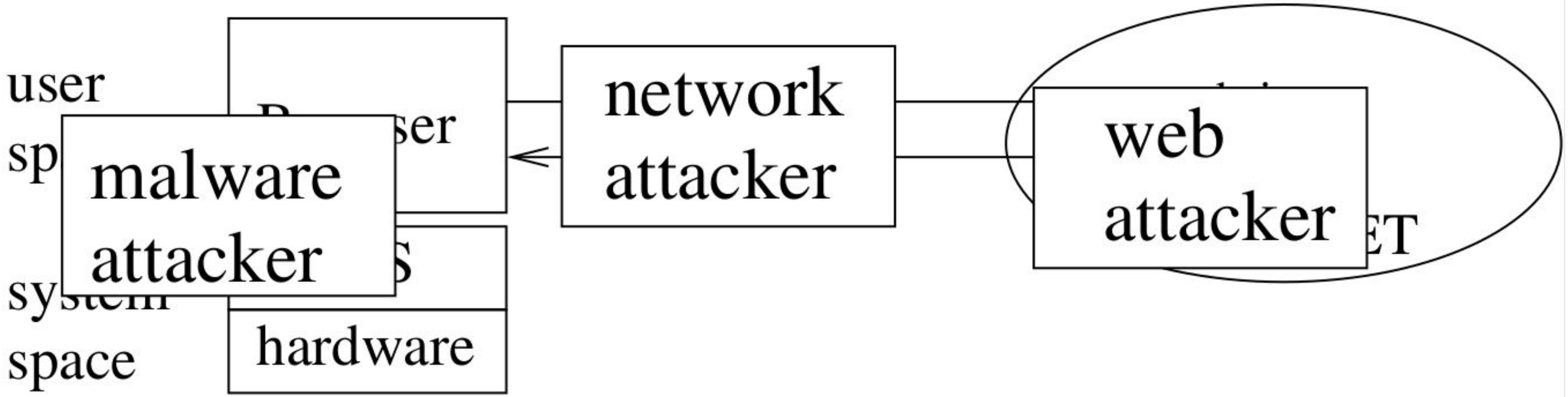
- web browser
  - responsible for securely confining Web content
  - e.g., evil site iframing a good site must not learn your password as you type it in
- web applications
  - e.g., online merchants, banks, blogs, collaborative editing
  - mix of server-side and client-side code
    - JavaScript, PHP, Ruby, Python, etc. on server side
    - mostly JavaScript on client side
  - many potential bugs: XSS, XSRF, SQL-injection (next lectures)
    - e.g., evil site must not be able to send queries on your behalf even if you logged in through an iframe
    - e.g., evil site should not give a script that looks like it came from someone else
    - e.g., evil user should not be able to change the price of an item when purchasing


user
space
system
space
Browser
OS
hardware
request
reply
website
INTERNET


user
space
malware
attacker
system
space
hardware
network
attacker
web
attacker

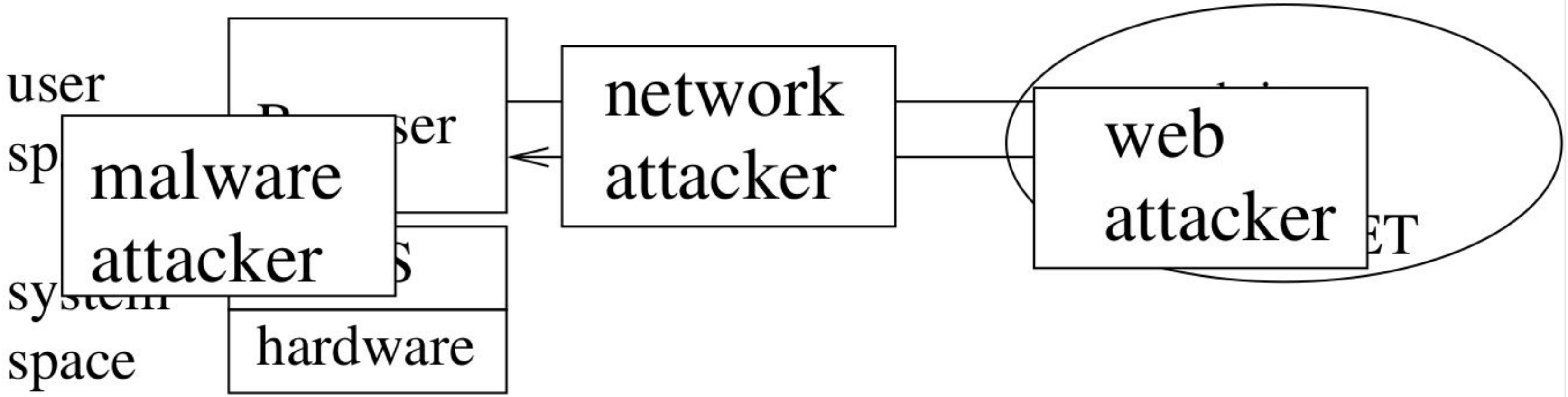
Malware Attacker

- malicious code runs on victim’s computer
- can exploit bugs in software
- can convince user to install malicious content
    - e.g., masquerade as anti-virus, video codec, etc.


user
space
system
space
malware
attacker
hardware
network
attacker
web
attacker
ET

Network Attacker

- passive attacks
  - wireless eavesdropper
- active attacks
  - evil Wi-Fi router
  - DNS poisoning


user
sp
malware
attacker
ser
S
system
space
hardware
network
attacker
web
attacker
ET

Web Attacker

- controls a malicious website (attacker.com)
  - can even have a nice SSL/TLS certificate for the site
- user visits attacker.com
  - phishing email
  - enticing content
    - 10 things that you should see, especially number 7!
- placed by an ad network
- clicked by accident
- tricked into clicking (clickjacking)
- attacker has no other access to user machine
  - only the content that it gives to the user

iframe attacker:

iframe with malicious content included
in otherwise honest webpage

iframe attacker:

iframe with malicious content included
in otherwise honest webpage

e.g., ads, analytics, trackers

iframe attacker:

iframe with malicious content included
in otherwise honest webpage

e.g., ads, analytics, trackers

come from third parties but appear
on the same page as first party

Browser: Basic Execute Model

- each browser window or frame
  - loads content
  - renders it to the screen
    - processes HTML, CSS, run javascript
    - load images, subframes, etc.
  - respond to events
- events
  - user actions like onclick, onkeydown
  - rendering behaviour like onload, error
  - timer elapsing like setTimeout
  - AJAX: dynamic loading of content

Document Object Model (DOM)

- HTML is internally represented as a `document` object
  - it has `properties` that are themselves objects
  - these are arranged in a hierarchical structure
  - everything in HTML is put somewhere inside
    - `anchor` objects have `href` properties
- the browser displays the HTML on a frame
  - can be a window or part of a window
  - represented by a `window` object
  - the `window.location` property has the URL components as properties
- the `document` object is the root and is standardized by the DOM
  - the DOM is an API for JavaScript to manipulate anything inside
  - e.g., `window.location = ‘‘evil.com’’`

JavaScript

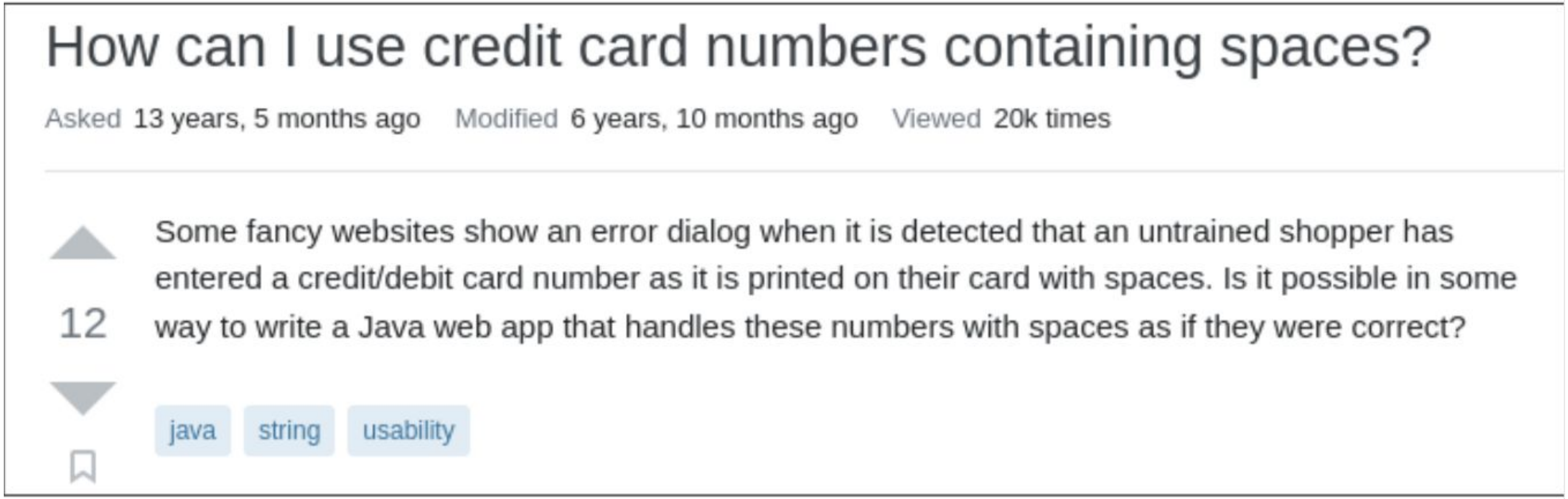
- language executed by the browser
  - scripts are embedded in webpages
  - can be set to run at different times:
    - before loading HTML
    - before viewing page
    - while viewing page
    - when leaving page
- JavaScript allows malicious webpages to execute code on user’s machine
  - you download code and run it

JavaScript History

- created by Brendan Eich at Netscape
    - scripting language for Navigator 2
- later standardized for cross-browser compatibility
    - ECMAScript
- has nothing to do with java
    - name was part of marketing deal
    - “Java is to JavaScript as car is to carpet”

Uses of JavaScript

- active and immersive web experiences!
- special effects
    - change images, hide elements, change cursor
- dynamic content manipulation
- form validation
    - “credit card field must not have spaces”
        - `LEAST SURPRISE FAILURE`
        - design failure
        - possible `INPUT VALIDATION FAILURE`
- lots of web apps
    - collaborative editing, social media, etc.



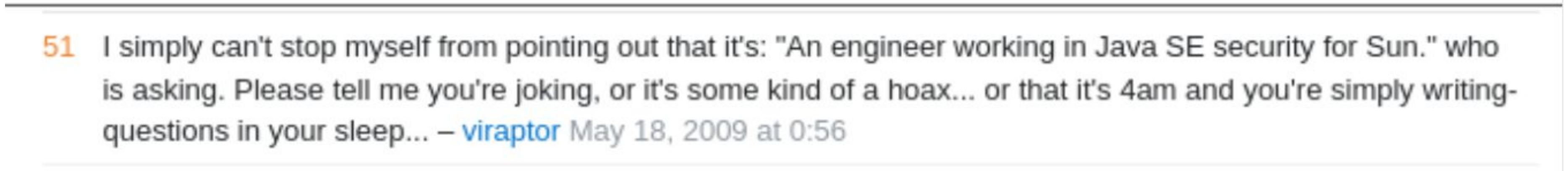
How can I use credit card numbers containing spaces?

Asked 13 years, 5 months ago Modified 6 years, 10 months ago Viewed 20k times

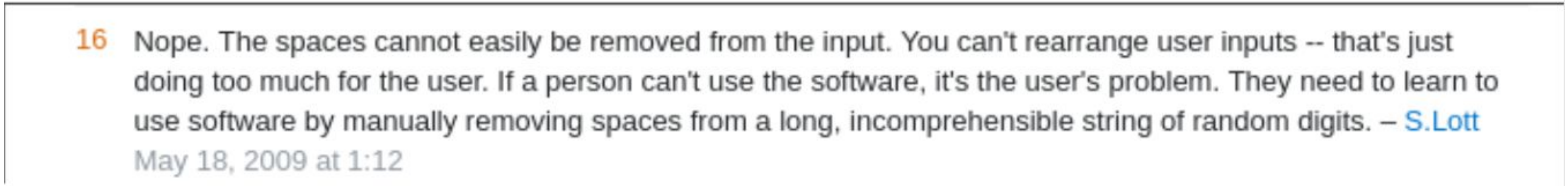
12

Some fancy websites show an error dialog when it is detected that an untrained shopper has entered a credit/debit card number as it is printed on their card with spaces. Is it possible in some way to write a Java web app that handles these numbers with spaces as if they were correct?

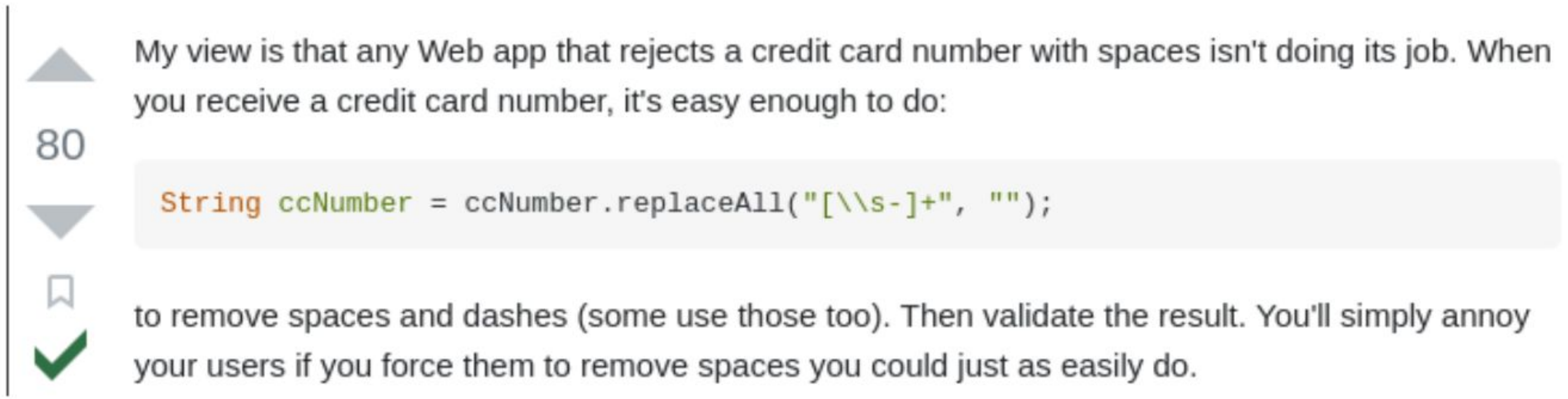
java string usability



51 I simply can't stop myself from pointing out that it's: "An engineer working in Java SE security for Sun." who is asking. Please tell me you're joking, or it's some kind of a hoax... or that it's 4am and you're simply writing-questions in your sleep... – viraptor May 18, 2009 at 0:56



16 Nope. The spaces cannot easily be removed from the input. You can't rearrange user inputs -- that's just doing too much for the user. If a person can't use the software, it's the user's problem. They need to learn to use software by manually removing spaces from a long, incomprehensible string of random digits. – S.Lott May 18, 2009 at 1:12



My view is that any Web app that rejects a credit card number with spaces isn't doing its job. When you receive a credit card number, it's easy enough to do:

```
String ccNumber = ccNumber.replaceAll("[\\s-]+", "");
```

to remove spaces and dashes (some use those too). Then validate the result. You'll simply annoy your users if you force them to remove spaces you could just as easily do.

Where to put JavaScript

- embedded in HTML as a <script> element
  - <script> alert(“Hello, world!”) </script>
  - <script type=“text/JavaScript” src=“notmalicious.js” />
- event handler in a tag
  - <a href=“http://www.tru.ca” onmouseover=“alert(‘click’);”>
- JavaScript schema
  - <a href=“JavaScript: alert(‘click’);”>click here</a>

JSON

- JavaScript Object Notation
    - serialization format for pure data
    - null, ints, floats, strings, arrays, and objects
    - dominant exchange format for HTTP
- object: {key: “value”, number: 2}
- array: [3, “hello”, 12.34, NaN, null, {point: [3, 6] }]

```
{
  "input": {
    "actor_id": "0",
    "app_use_state": "FOREGROUND",
    "bluetooth_info": {
      "enabled": "true",
      "scan_results": [
        {
          "advertisement_payload_base64": "0201061bff0118beac626974657962697465796269746579626974657 9f401",
          "age_ms": "5548",
          "hardware_address": "AB:B1:E7:7E:1B:BA",
          "rssi_dbm": "-12",
        },
        {
          "advertisement_payload_base64": "0201061aff4c00021501022022fa0f010000acdd1c6502da1cd0e7a64bf4",
          "age_ms": "5495",
          "hardware_address": "AB:B1:E6:6E:1B:BA",
          "rssi_dbm": "-12",
        },
        {
          "advertisement_payload_base64": "0201060303aafe1116aafe10f4006c6f6f73656c6174636807",
          "age_ms": "5445",
          "hardware_address": "AB:B1:E8:8E:1B:BA",
          "rssi_dbm": "-12",
        },
        {
          "advertisement_payload_base64": "0201061bff0118beac626974657962697465796269746579626974657 9f401",
          "age_ms": "5343",
          "hardware_address": "AB:B1:E3:3E:1B:BA",
          "rssi_dbm": "-12",
        },
      ],
    },
    "cell_info": {
      "connected": [
        {
          "is_network_roaming": "false",
          "network_country_iso": "ca",
```

Same Origin Policy (SOP)

Same Origin Policy (SOP)

SOP is an isolation and access control philosophy to protect data connected to one host being accessed by another (possibly malicious) host

Same Origin Policy

- avoid having an evil website:
  - read data from another website
  - manipulate data from another website
- SOP is a compromise
  - one option would be to not allow any content other than the website itself
    - e.g., embed all files and scripts into the single HTML page
  - another would be to not allow any third-party content
  - SOP allows third-party content, but sandboxes it
    - ideally treats it as though it were another opened tab
    - considered a vulnerability when not true
- SOP controls all access to the DOM

Same Origin Policy

- a base HTML document is assigned an origin
  - based on the URI that retrieved it
  - consists of (scheme, host, port) triple
    - scheme://host:port/irrelevant/to/sop
  - two origins are the same if the whole triple matches
    - string matching

Same Origin?

Same Origin?

http://wikipedia.org/a/ and http://wikipedia.org/b/

Same Origin?

http://wikipedia.org/ and http://www.wikipedia.org/

Same Origin?

http://wikipedia.org/ and https://wikipedia.org/b/

Same Origin?

http://wikipedia.org:81/ and http://wikipedia.org:82/

Same Origin?

http://wikipedia.org/a/ and http://wikipedia.org/b/

http://wikipedia.org/ and http://www.wikipedia.org/

http://wikipedia.org/ and https://wikipedia.org/b/

http://wikipedia.org:81/ and http://wikipedia.org:82/

SOP assigns every component in DOM an origin based on what loaded it into the webpage

SOP assigns every component in DOM an origin based on what loaded it into the webpage

images, scripts, content get the origin of loader

SOP assigns every component in DOM an origin
based on what loaded it into the webpage

images, scripts, content get the origin of loader

the content of an iframe has the
origin of the URL that serves the iframe;
not the website that embeds it.

SOP: one origin should not be able to access the resources of another origin.

SOP: one origin should not be able to access the resources of another origin.

JavaScript on one iframe cannot read or modify pages from different origins

E.g., consider a website that has an
iframe that loads a dynamic website
whose content changes based on IP

E.g., consider a website that has an iframe that loads a dynamic website whose content changes based on IP

e.g., library, or a LAN resource

E.g., consider a website that has an iframe that loads a dynamic website whose content changes based on IP

e.g., library, or a LAN resource

should this website be able to read that?

E.g., consider a website that has an
iframe that loads a dynamic website
whose content changes based on IP

e.g., library, or a LAN resource

should this website be able to read that?

or modify how it looks?

Web Cookies

HTTP is Stateless

HTTP is Stateless

each HTTP request is a standalone event

HTTP is Stateless

each HTTP request is a standalone event

this is a poor match to how it is used

HTTP is Stateless

each HTTP request is a standalone event

this is a poor match to how it is used

e.g., shopping carts, being “logged-in”, language preferences

Web cookies add state

Web cookies add state

a cookie is a file created by a website to store information in the browser

Web Cookies

- server replies with an HTTP header Set-Cookie
  - e.g., JSESSIONID=4F5627840794D2E93F9D083EDFD9F263; Path=/; HttpOnly
- cookie has parameters
  - “argument=value” format or just “argument”
  - arguments are optional
  - arguments are separated by semicolons

SOP for Cookies

- cookies have a different SOP than the DOM’s SOP
- can be sent on any port
- can be sent using HTTP or HTTPS
- must be sent to same domain and path
    - subdirectories okay
- these can be tweaked

Web Cookie Arguments

- Secure; (only send over encrypted channel)
- HttpOnly; (deny DOM API access to cookie)
- Expires=Mon, 29-Jan-2018 14:30:11 GMT; (date to delete)
    - format is Wdy, DD-Mon-YYYY HH:MM:SS GMT
    - past date means delete immediately
    - no expire means this session only

Web Cookie Arguments

- Domain=VALUE
  - by default cookie can only be read by exact domain
    - e.g., myserver.eng.tru.ca
  - is used to widen domain
    - e.g., .tru.ca
    - means anything that ends in .tru.ca
    - is this secure? what is threat model?
  - cannot be used to allow different domain
- Path=VALUE;
  - widen path of who may read the cookie
  - by default only HTML in the current directory and subdirectories can get cookie

Cookie Uses

- session ID
  - random number stored in a cookie and sent to the server
  - indexes server-side state for the client
- authentication
  - cookie's session ID also acts as an authenticator
    - the cookie proves to the website that the client previously authenticated correctly
  - what does this remind you of?
  - cookie theft: anyone knowing this number can impersonate you
    - why can't it prove that it is Alice?

SOP motivation

SOP motivation

if a script loads a website in an iframe
it may be appear different based on cookie

SOP motivation

if a script loads a website in an iframe
it may be appear different based on cookie

e.g., it is logged into to website

SOP motivation

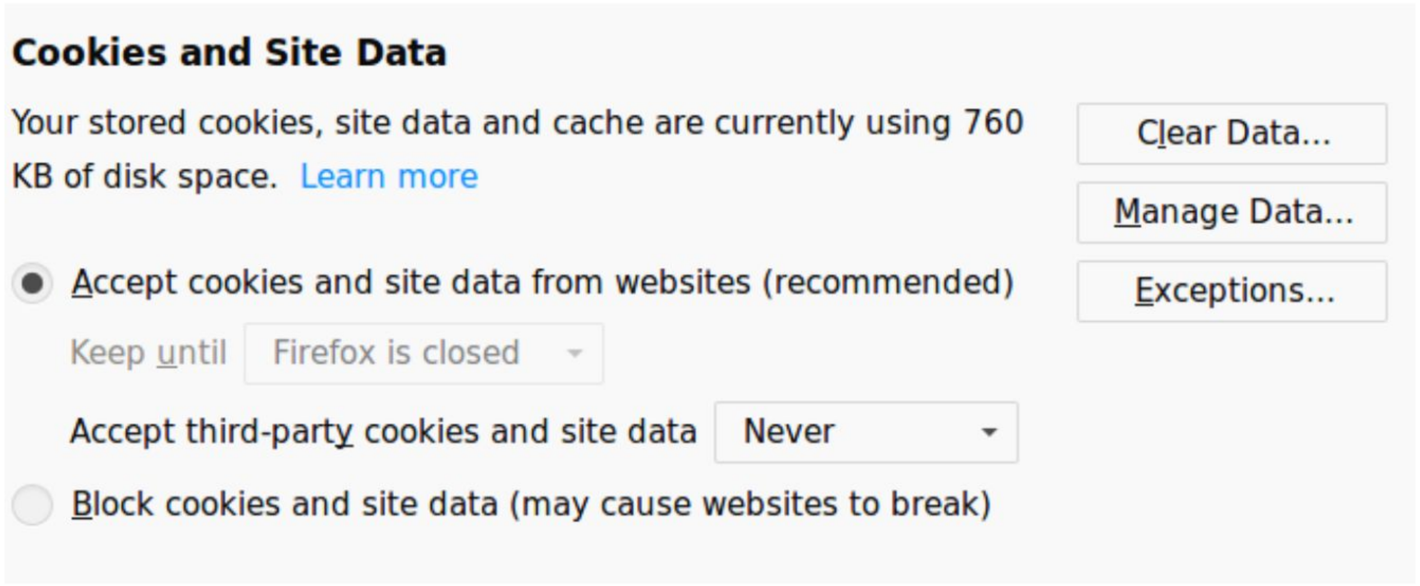
if a script loads a website in an iframe
it may be appear different based on cookie

e.g., it is logged into to website

SOP prevents script from reading that iframe

Cookie Uses

- personalization
    - helps the website recognize the user from a previous visit
    - store settings locally or remotely
- tracking
    - follow the user from site to site
    - learn their browsing behaviour, etc.
- third-party cookies
    - a.com has an iframe for b.com
    - b.com sends a cookie for the site
    - user will send that back to b.com whenever visiting a.com



Cookies and Site Data

Your stored cookies, site data and cache are currently using 760 KB of disk space. Learn more

● Accept cookies and site data from websites (recommended)

Keep until: Firefox is closed

Accept third-party cookies and site data: Never

○ Block cookies and site data (may cause websites to break)