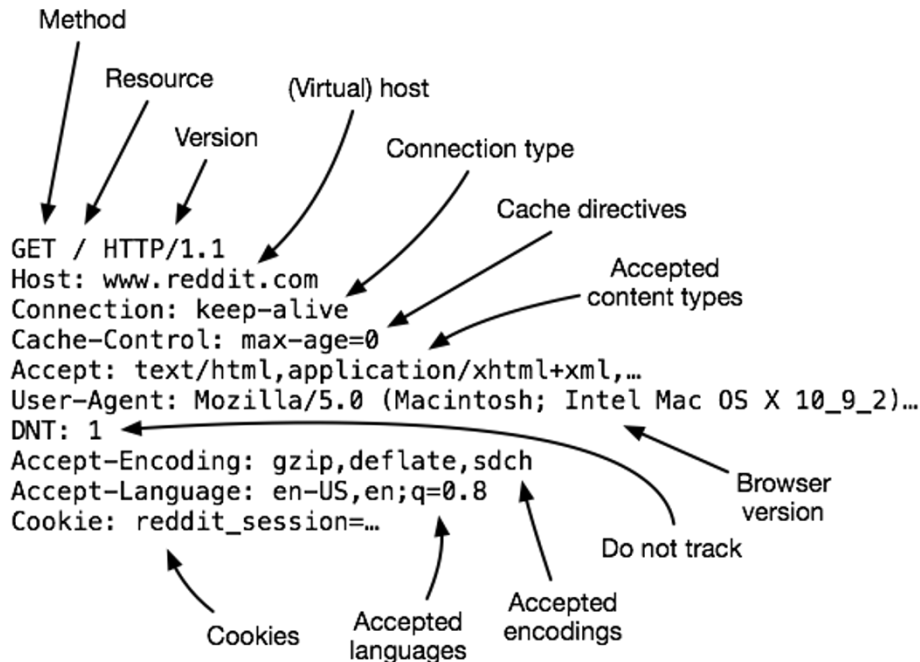


**CS 88: Week 4: Class 8: Web Security: HTTP and Cookies**  
**Discussion Question 1: HTTP Headers**

**Part A:** Understanding HTTP Request-Response Headers.

HTTP is the Hypertext Transfer Protocol. The headers for an HTTP request are shown below:



# Anatomy of Request

## HTTP Request

**method**      **path**      **version**  
GET    /index.html    HTTP/1.1

```
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Host: www.example.com
Referer: http://www.google.com?q=dingbats
```

headers

[Empty box representing the body of the request]

body  
(empty)

There are many ways to send data over HTTP. Match the following HTTP request types, with their corresponding syntax.

Four ways to send data to the server

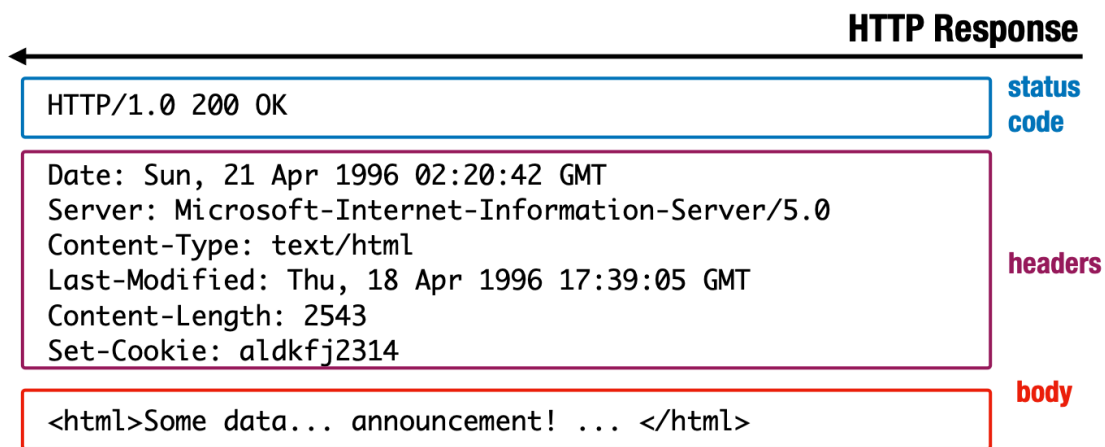
1. Embedded in the URL (typically URL encoded, but not always)
2. In cookies (cookie encoded)
3. Inside a custom HTTP request header
4. In the HTTP request body

Examples

- a. GET /purchase.html?user=alice&item=iPad&price=400 HTTP/1.1
- b. Cookie: user=alice; item=iPad; price=400;
- c. BODY of HTTP POST user=alice&item=iPad&price=400
- d. My-Custom-Header: alice/iPad/400

Let's say a website decided to use Example (C) to send the price of an iPad to Alice. Alice wants to buy the iPad but thinks the price is exorbitant. Given the anatomy of a response shown below, is it possible for Alice to buy the iPad for \$0?

# HTTP Response



Part B: Same-Origin Policy: Select whether the following websites have the same origin.

## Same-Origin Policy

- Two webpages have the same origin *if and only if* the protocol, domain, and port of the URL all match exactly: string matching:
  - The **protocol**, **domain**, and **port** strings must be equal

First domain	Second domain	Same origin?
<code>http://cs88.swat.org</code>	<code>https://cs88.swat.org</code>	
<code>http://cs88.swat.org</code>	<code>http://swat.org</code>	
<code>http://cs88.swat.org</code> <code>[:80]</code>	<code>http://cs88.swat.org:8000</code>	

The Same-origin policy has some exceptions:

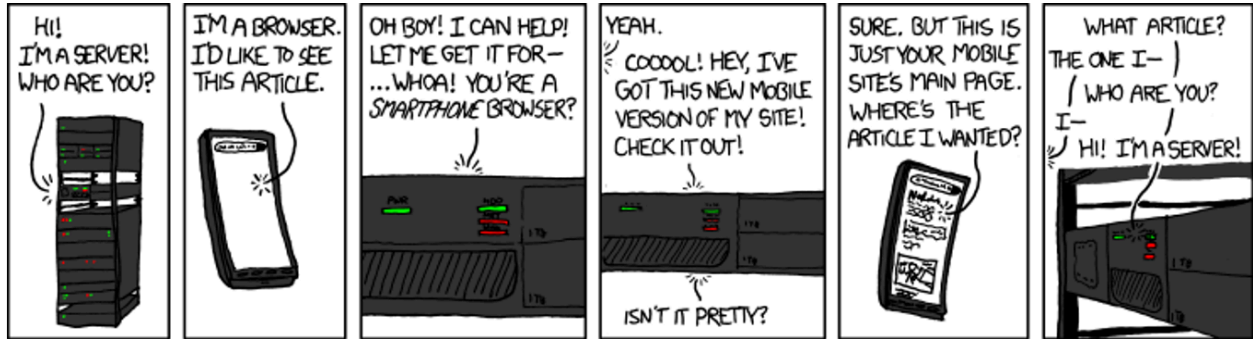
- JavaScript runs with the origin of the page that loads it
- Websites can fetch and display images from other origins
- Websites can agree to allow some limited sharing

Having learnt about Same-origin policy, you are asked to provide insight into the security vulnerabilities for the webpage `cs88.com`.

The webpage `cs88.org` embeds `google.com`. We know that because of the same origin policy, the inner frame for `google.com` cannot interact with the outer frame for `cs88.org` and vice-versa. Given this information, what happens when:

- `cs88.org` fetches Javascript from Google analytics.
- `cs88.org` includes `<imgsrc="http://google.com/logo.jpg">` and the image has origin `http://google.com`.
- An iframe `<imgsrc="http://google.com/logo.jpg">` is loaded to the `cs88.org` webpage and the image has origin `http://google.com`.

## Discussion Question 2: Cookies



**Part A:** Introducing state into HTTP. We said that HTTP is stateless. I.e., every time you talk to the same server, it forgets all past interactions. List three reasons why we need state when communicating with a server.

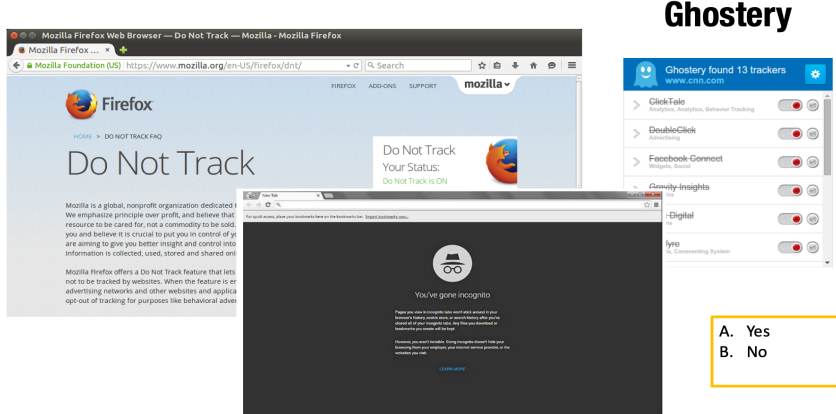
## Part B: Detecting Cookies

How many cookies do you see on this page?

A screenshot of the Los Angeles Times website. The page features a navigation bar with "TOPICS", "SEARCH", "LOCAL", "POLITICS", "SPORTS", "ENTERTAINMENT", "OPINION", and "PLACE AN AD". There are promotional banners for "GO UNLIMITED!" and "4 WEEKS FOR ONLY 99¢". The main headline is "Islamic State claims it was behind Sri Lanka bombings" by Shashank Bengali. Other news items include "As Coachella raged, the L.A. tech world..." and "Beware of late-night lane closures on your way to (and from) LAX". A "Casper" advertisement is also visible. At the bottom, there is a "TRIAL OFFER" for "SUPPORT QUALITY NEWS" and a "Cookie Notice" stating: "By continuing to use our site, you agree to our Terms of Service and Privacy Policy. You can learn more about how we use cookies by reviewing our Privacy Policy. Close".

As a privacy conscious web consumer, you decide to turn on Do Not Track in your browser. Will this ensure you are not tracked?

If you use Do Not Track in your browser, will that ensure that no third-party cookies are set?



The image shows a Firefox browser window with the 'Do Not Track' feature enabled. The status bar indicates 'Do Not Track is ON'. To the right, the Ghostery extension interface is visible, showing it has found 13 trackers. Below the browser window, a yellow box contains the following options:

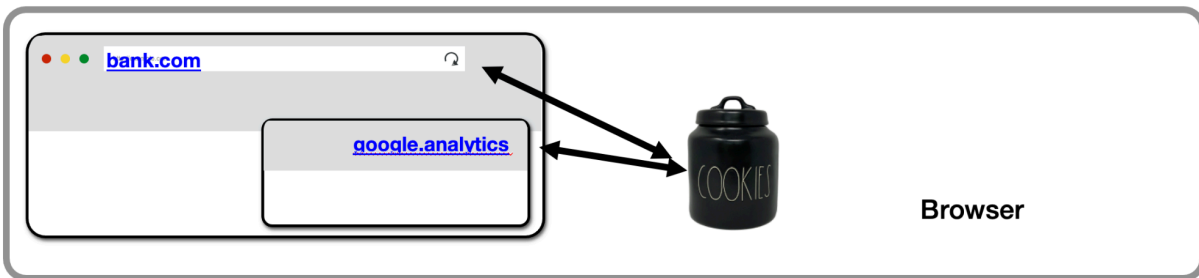
A. Yes  
B. No

- A. Yes (Explain why)
- B. No (Explain why)

**Part C: Cookie Policy:** Cookie policies are a set of rules enforced by the browser

- When the browser receives a cookie from a server, should the cookie be accepted?
- When the browser makes a request to a server, should the cookie be attached?
- **Cookie policy is not the same as same-origin policy**

Can the following attack succeed?



If we have a google analytics Javascript running on bank.com's login page. Assume that the site has no frames, and everything on this page has the same origin. Can google analytics see Alice's session cookie on bank.com?


- A. Yes
- B. No
- C. Maybe
- D. Something Else

## Part D: Setting Cookie Policies

### Cookie Policy: Setting Cookies


- When the browser receives a cookie from a server, should the cookie be accepted?
- Server with **domain X** can set a cookie with **domain attribute Y** if
  - The **domain attribute** is a **domain suffix** of the **server's domain**
    - X ends in Y
    - X is below or equal to Y on the hierarchy
    - X is more specific or equal to Y
  - The **domain attribute Y** is not a top-level domain (TLD)
  - No restrictions for the Path attribute (the browser will accept any path)
- Examples:
  - [mail.google.com](mailto:mail.google.com) can set cookies for Domain=[google.com](https://www.google.com)
  - [google.com](https://www.google.com) can set cookies for Domain=[google.com](https://www.google.com)
  - [google.com](https://www.google.com) **cannot** set cookies for Domain=[com](https://www.com), because com is a top-level domain

### Cookie Policy: Sending Cookies

(server URL)  
<https://cs88.swat.edu/cryptoverse/oneshots/subway.html>  
[cs88.swat.edu/cryptoverse](https://cs88.swat.edu/cryptoverse)   
(cookie domain) (cookie path)

Quick method to check cookie sending:  
Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.

If the domains and paths all match, then the cookie is sent.

(server URL)  
<https://cs88.swat.org/cryptoverse/oneshots/subway.html>  
[cs88.swat.org/xorcist](https://cs88.swat.org/xorcist)   
(cookie domain) (cookie path)

# Scoping Example

name = cookie1  
value = a  
domain = login.site.com  
path = /

name = cookie2  
value = b  
domain = site.com  
path = /

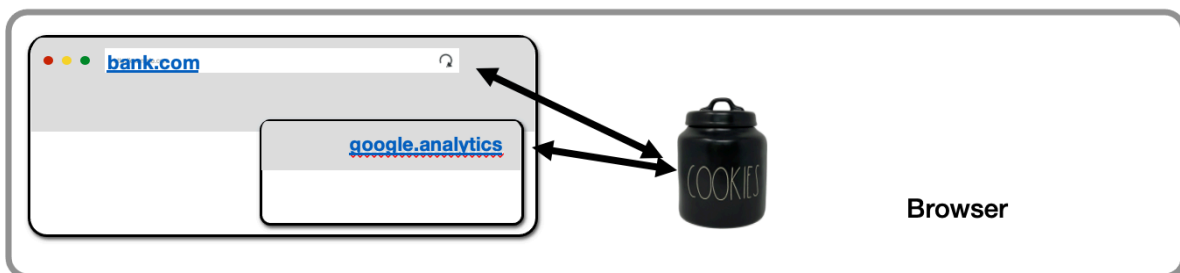
name = cookie3  
value = c  
domain = site.com  
path = /my/home

cookie domain is suffix of URL domain  $\wedge$  cookie path is a prefix of URL path

	Cookie 1	Cookie 2	Cookie 3
<a href="http://checkout.site.com">checkout.site.com</a>			
<a href="http://login.site.com">login.site.com</a>			
<a href="http://login.site.com/my/home">login.site.com/my/home</a>			
<a href="http://site.com/account">site.com/account</a>			

Given what we just learnt, can the following attack succeed?

## Can the following attack succeed?



If we have a google analytics Javascript running on bank.com's login page. Assume that the site has iframes. Can google analytics see Alice's session cookie on bank.com?