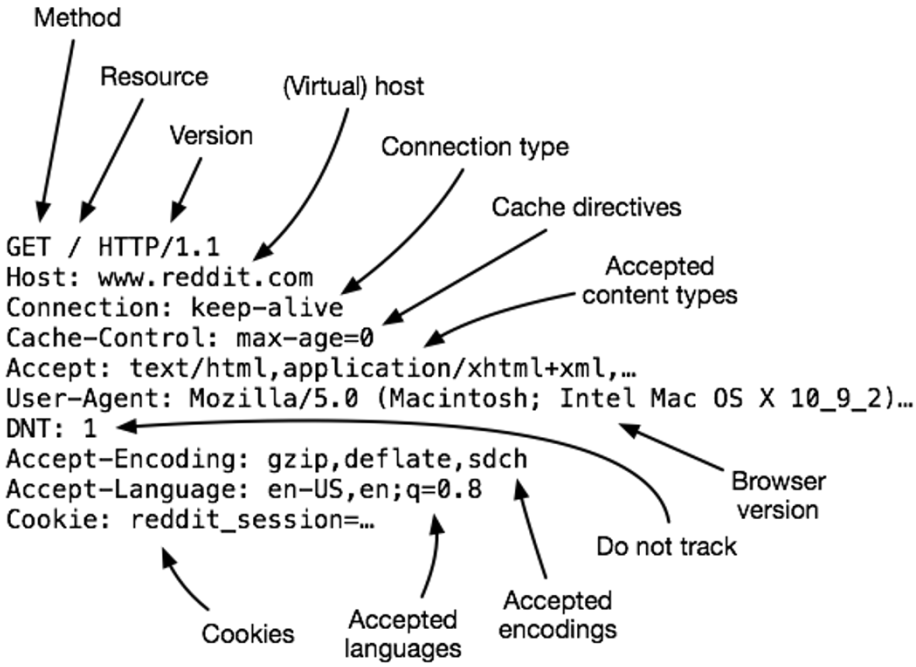
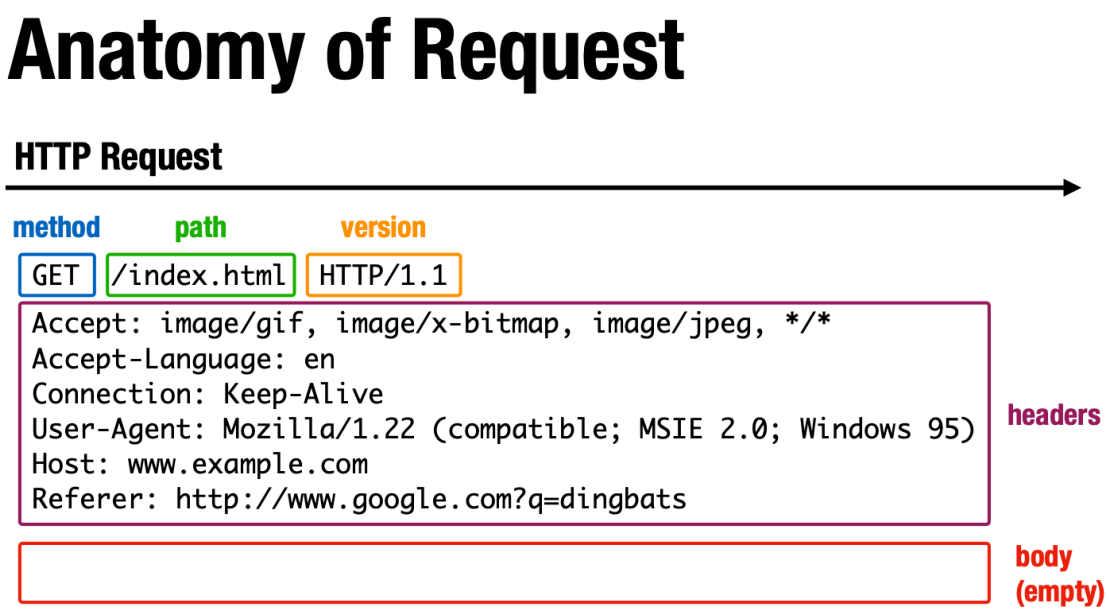
CS 88: Week 4: Class 8: Web Security: HTTP and Cookies
Discussion Question 1: HTTP Headers

Part A: Understanding HTTP Request-Response Headers.
HTTP is the Hypertext Transfer Protocol. The headers for an HTTP request are shown below:





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

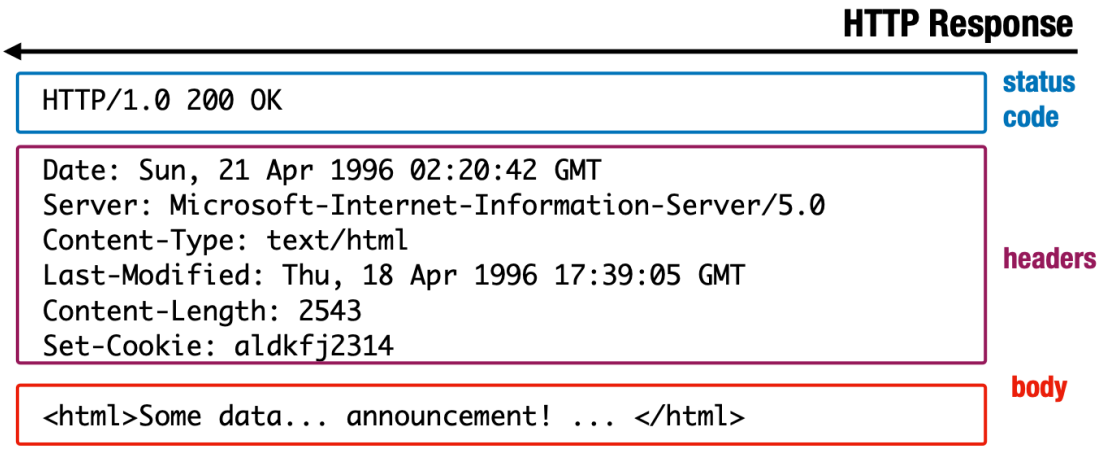
1. Embedded in the URL (typically URL encoded, but not always)
2. In cookies (cookie encoded)
3. Inside a custom HTTP request header
4. In the HTTP request body

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 want's 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?
`http://cs88.swat.org`	`https://cs88.swat.org`	
`http://cs88.swat.org`	`http://swat.org`	
`http://cs88.swat.org` [:80]	`http://cs88.swat.org:8000`	

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.

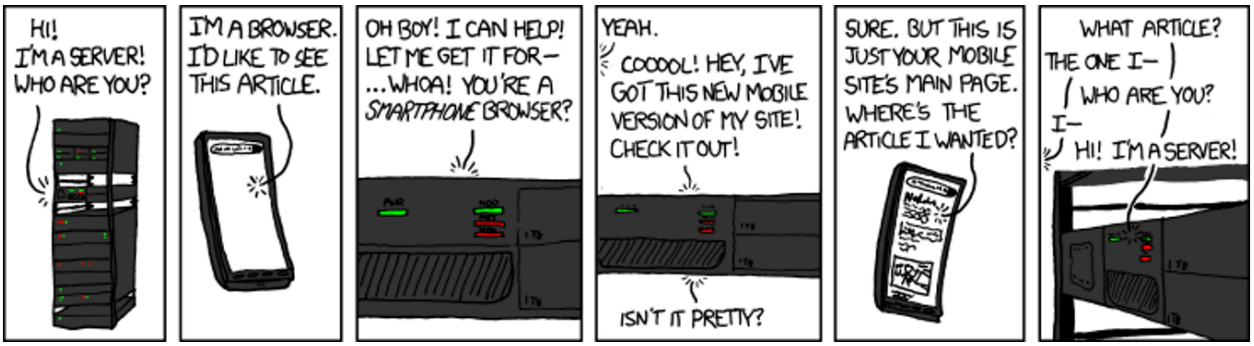
The webpage `cs88.org` embeds google.com. We know that because of the same origin policy, the inner frame for `google.com` cannot interact with the outer frame for `cs88.org` and vice-versa. Given this information, what happens when:

A. `cs88.org` fetches Javascript from Google analytics.

B. `cs88.org` includes `<imgsrc="http://google.com/logo.jpg”>` and the image has origin `http://google.com`.

C. 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

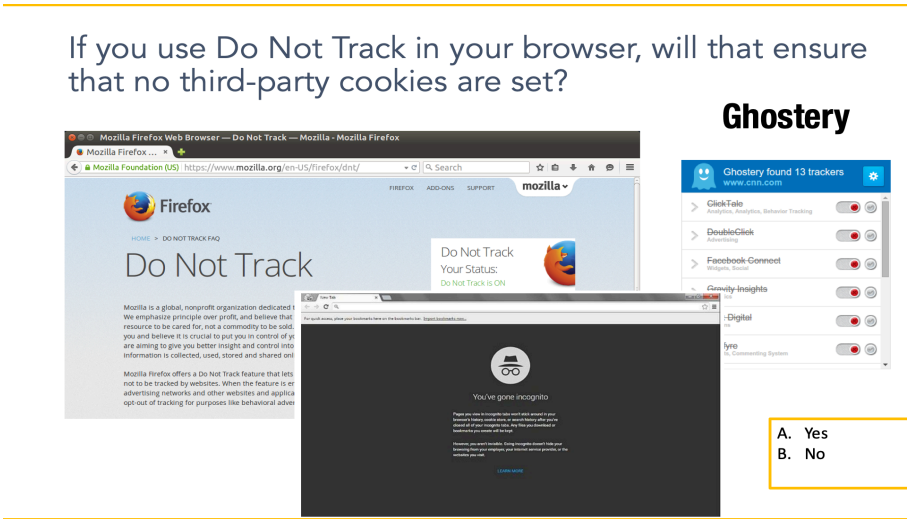


Part A: Introducing state into HTTP. We said that HTTP is stateless. I.e., every time you talk to the same server, it forgets all past interactions. List three reasons why we need state when communicating with a server.

Part B: Detecting Cookies

How many cookies do you see on this page?

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

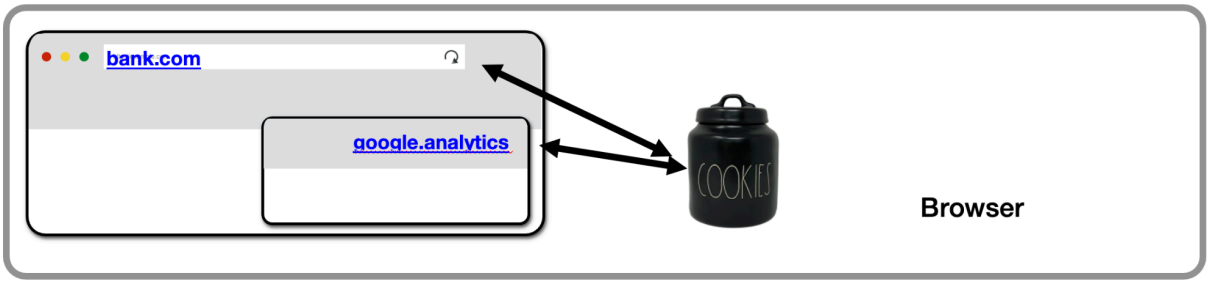


A. Yes (Explain why)
B. No (Explain why)

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?
- **<u>Cookie policy is not the same as same-origin policy</u>**

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** can set cookies for Domain=google.com
  - **google.com** can set cookies for Domain=google.com
  - google.**com** **cannot** set cookies for Domain=com, because com is a top-level domain

33

Cookie Policy: Sending Cookies

(server URL)
`https://cs88.swat.edu/cryptoverse/oneshots/subway.html`

`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` ❌
(cookie domain) (cookie path)

Scoping Example

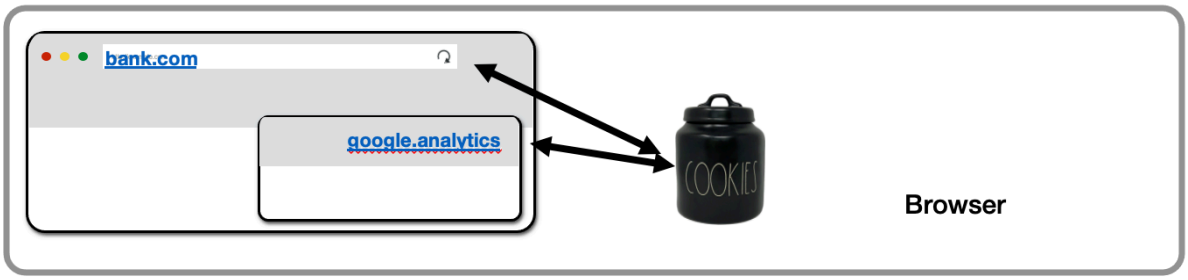
name = cookie1 value = a domain = login.site.com path = /	name = cookie2 value = b domain = site.com path = /	name = cookie3 value = c domain = site.com path = /my/home

cookie domain is suffix of URL domain ∧ cookie path is a prefix of URL path

	Cookie 1	Cookie 2	Cookie 3
checkout.site.com			
login.site.com			
login.site.com/my/home			
site.com/account			

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?