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

```
Method
Resource
(Virtual) host
Version
Connection type
Cache directives
Accepted
content types
Accept: text/html,application/xhtml+xml,...
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2)...
DNT: 1
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US, en; q=0.8
Cookie: reddit_session=...

Method
Resource
(Virtual) host
Version
Connection type
Cache directives
Accepted
content types
Accept: text/html,application/xhtml+xml,...
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2)...
DNT: 1
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US, en; q=0.8
Cookie: reddit_session=...

Anatomy of Request

HTTP Request

<table>
<thead>
<tr>
<th>method</th>
<th>path</th>
<th>version</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/index.html</td>
<td>HTTP/1.1</td>
</tr>
</tbody>
</table>

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

(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

HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543
Set-Cookie: aldkfj2314

<html>Some data... announcement! ... </html>
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

<table>
<thead>
<tr>
<th>First domain</th>
<th>Second domain</th>
<th>Same origin?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http://cs88.swat.org</code></td>
<td><code>https://cs88.swat.org</code></td>
<td></td>
</tr>
<tr>
<td><code>http://cs88.swat.org</code></td>
<td><code>http://swat.org</code></td>
<td></td>
</tr>
</tbody>
</table>

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.
C. An iframe `<img src="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?

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

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
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

Cookie Policy: Sending Cookies

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

`cs88.swat.edu/cryptoverse`  (cookie domain)  `subway.html`  (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/onshots/subway.html

`cs88.swat.org/xorclist`  (cookie domain)  `subway.html`  (cookie path)

×
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?