Problem 1 The developers of the Zoobar social networking site would like to add a new page to allow logged-in users to see their username and account balance (which is considered secret in Zoobar 2.0). They first create a template page, which is the same for all users:

```
<html><body>
<h1>Your Zoobar Info</h1>
<p>Your username is <span id="user"></span> and your balance is $<span id="balance"></span>.</p>
<script>
var setname = function(name) {
  document.getElementById("user").textContent = name;
}
var setbalance = function(dollars) {
  document.getElementById("balance").textContent = dollars;
}
</script>
<!-- Fetch user info as JSON and render it -->
<script src="/userinfo.js"></script>
</body></html>
```

For a logged-in user, the `http://zoobar.org/userinfo.json` endpoint will return a personalized JSON file, e.g.,

```
{"username": "alice", "balance": 20}
```

A consultant brought in by Zoobar's VCs suggests that they reduce load time by replacing lines 11–21 of the page above with just

```
<!-- Fetch user info in JavaScript -->
<script src="/userinfo.js"></script>
```
Now, for a logged-in user, the `http://zoobar.org/userinfo.js` endpoint will return a personalized JavaScript file, e.g.,

```javascript
setname("alice"); setbalance(20);
```

This is a very bad idea for the privacy and security guarantees that the Zoobar site is supposed to provide. Explain why. Be specific.

**Problem 2** Prior to release 42 of the Google Chrome browser, users could choose one of three settings for content rendered by plugins such as Flash: always enabled; click-to-play; and always disabled. Under the “always enabled” setting, Flash content embedded in Web page played automatically. This was the default. Under “click-to-play,” Flash content began playing once its placeholder was clicked; see the screenshot below, left. Even under “always disabled,” it was possible to get Flash content to play, by right-clicking the placeholder and choosing “Run this Plugin” from the context menu; see the screenshot below, right.

For security, the developers of Chrome strongly recommended “always disabled” and right clicking to enable Flash content, rather than “click-to-play,” even assuming that that users will in either case allow certain Flash content (such as videos or games) to run.

**Why is “always disabled” more secure than “click-to-play”?**

**Problem 3** To simplify certificate deployment, TLS supports wildcard certificates. For example, many of UCSD’s TLS servers use a single wildcard cert for “*.ucsd.edu”:

---

1Indeed, as of release 42 of Chrome, in April of last year, “click-to-play” is gone, replaced with a new option, “detect and run important plugin content,” which tries to run plugins that constitute the main page content, while applying the “always disabled” policy to ancillary plugins like those used in banner ads.
Suppose that https://blink.ucsd.edu/, a server used by the University for many sensitive HR applications (IP: 132.239.180.101), and https://cseweb.ucsd.edu/, a server that hosts home pages for CS department members (IP: 132.239.8.67), both present UCSD’s wildcard cert for TLS connections. Suppose further that the Apache server installed on https://cseweb.ucsd.edu/ is configured to support only that virtual host, and will serve content from the cseweb.ucsd.edu site regardless of what HTTP Host: header the client sends.  

Alice Attacker is a graduate student in the CSE department, and has a home page at https://cseweb.ucsd.edu/~attacker/, where she can place arbitrary content. In addition, Alice Attacker has set up a rogue UCSD-GUEST access point on campus, allowing her to act as an in-path network attacker against any user who connects to that access point.

**Explain how Alice can completely undermine origin isolation for logged-in users of https://blink.ucsd.edu/ who connect to her access point, by injecting JavaScript of her choice into the https://blink.ucsd.edu origin in their browsers.** Be specific.

---

This is not the actual configuration of these servers. Do not attempt to attack blink.ucsd.edu, or any other server, without permission from its owners.