Collaboration policy reminder: You may discuss a homework assignment with one other student in the class. You must write up your solutions separately. If you discussed the solutions with anyone, please note so on your solutions. You must not look at homework, programming project, or exam solutions from previous years of this class, or equivalent classes at other schools.

You may use online resources for general reference, but not to search for solutions to specific questions posed in this homework. For example, Googling for the HTTPS protocol specification is okay; Googling specifically for “why the 502 error code is in plaintext” is not allowed. If you are unsure about whether a use is allowed, check first with the professor or TA.

Problem 1 Many corporate environments interpose a Web proxy between clients and the Web. These proxies can cache popular pages, reducing bandwidth use, and can inspect and log the requests made by user to ensure that they comply with corporate policy. Clients can be instructed what proxy to use by means of protocols like DHCP.

The HTTP protocol is self-proxying: the client connects to the proxy as though it were the server and makes the request. The proxy connects to the appropriate server, fetches the resource, and responds to the clients. (One reason that requests include a Host header is so proxies know what server to connect to.) This is illustrated in Figure 1.

Web proxies cannot be used in the same way to proxy HTTPS traffic. Instead, the client uses the CONNECT method to tell the proxy it would like to establish a connection to an HTTPS server. The proxy attempts to open a TCP connection to that server and port and, if successful, notifies the client. From that point on, the client and server communicate directly, with the proxy acting as a dumb intermediary. This is illustrated in Figure 2.

If the proxy is not able to connect to the HTTPS server, for example because DNS resolution fails, it notifies the client with an HTTP error code and a body (in plaintext) explaining what went wrong, which the browser can display to the user. This is illustrated in Figure 3.

1. Why is HTTPS proxying different from HTTP proxying? Why doesn’t the proxy connect to the server (using HTTPS), request the resource, and send that resource to the client, just as in the HTTP case?

2. In the case that the proxy is not able to establish a TCP connection to the server, why is its 502 error page sent to the client in plaintext?
Figure 1: A client makes an HTTP request for target.com through a Web proxy.

Figure 2: A client establishing an HTTPS connection to target.com through a Web proxy. After the 200 response, the proxy ferries the HTTPS traffic unchanged between the client and server.

3. Suppose that the 502 error page sent by the proxy includes JavaScript, and assume, for the purposes of this question, that this script will be executed in the target site’s origin. How can a malicious proxy use this to violate the security guarantees of the HTTPS protocol? Be specific and precise.

Problem 2 An HTTP server can set a cookie in a browser through the Set-Cookie header. This header looks like this:

HTTP/1.1 200 OK
...
Set-Cookie: name=value; Domain=.foo.com; Max-Age=3600

Other attributes can be applied to the cookie, including the Secure attribute considered below.
Figure 3: A client attempting to establish an HTTPS connection to a nonexistent server through a Web proxy. The proxy’s response gives an error code and an explanatory HTML page to display to the user.

The cookie will be sent on all subsequent connections (for up the next Max-Age seconds) to servers for which Domain is a suffix; for example, www.intranet.foo.com but not yafoo.com. If the Domain attribute is omitted, most browsers will send the cookie only to the host that set it. The cookie name-value pair is sent by the browser unaccompanied by any attributes such as Domain.

A server is allowed to set the Domain attribute of a Set-Cookie header to any suffix of its own hostname.¹

Browsers store cookies until they expire or the user’s cookie file grows too large; in the latter case, older cookies are expired to make room for newer ones.

Since cookies are used for state management on the Web, we would like the cookie mechanism to provide both confidentiality and integrity. Confidentiality here means that unauthorized servers should not be able to read a cookie set by foo.com; integrity here means that unauthorized servers should not be able to set cookies that will be sent by the browser to foo.com.

a. Against a Web attacker, do cookies set using the Set-Cookie header provide confidentiality? Do they provide integrity?

b. Suppose the Web attacker is additionally allowed to control servers whose hostname is a neighbor of the victim’s. For example, suppose that the attacker targeting www.foo.com controls the server at analytics.foo.com. Against this neighbor attacker, do cookies provide confidentiality? Do they provide integrity?

c. If the Set-Cookie header specifies the Secure attribute for a cookie, that cookie will only be sent from the browser to the server over the https: scheme, never over the http: scheme.

Against a network attacker, do cookies set using the Set-Cookie header with the Secure attribute provide confidentiality? Do they provide integrity?

¹But not to a “public suffix” shared by multiple users, such as .com.
Problem 3 The Like button is a widely deployed feature of the Facebook platform. A user who is logged in to Facebook can click on the Like button on a page she visits to express approval of the page or its content. This approval is reflected on that user’s news feed on the Facebook site. The suggested code for implementing a Like button on a page is:

```
<iframe
    scrolling="no" frameborder="0"
    style="border:none; width:450px; height:80px"></iframe>
```

a. Why is the Like button implemented in an iframe, rather than within the enclosing page itself?

b. Is clickjacking a concern for the Like button? What is the worst attack you can think of that is enabled by clickjacking Facebook Like buttons?

c. Should the JavaScript code making up the Like button include framebusting code to prevent clickjacking attacks?

Problem 4 Below is a picture of a UCSD police car parked near the Hopkins Parking Structure on campus. There is no one in the car.

![UCSD police car parked near the Hopkins Parking Structure on campus](image)

Why was this police car parked here? Explain. Be sure to specify the assumptions your answer makes about police behavior and driver behavior.

(Note: This question is not about computer security. Please don’t discuss computer security in your answer.)