CS161 Fall 2026 — Introduction to Computer Security — Discussion 9

Q1 *CSRF Trivia* **(4 points)**

Mark the following statements as True or False and justify your solution. Please feel free to discuss with students around you.

Q1.1 (1 point) CSRF requires the attacker to steal the victim's session cookie in order to succeed.

○ True ● False

Solution: CSRF relies on the browser automatically attaching the victim's existing valid session cookie to the forged request.

Q1.2 (1 point) JavaScript is **NOT** required to execute a CSRF attack on the attacker's malicious page.

● True ○ False

Solution: CSRF attacks can also be executed on URLs with malicious actions. For example, a GET request to `https://bank.com/transfer?amount=100&recipient=mallory` with a valid session token might send $100 to Mallory.

Q1.3 (1 point) CSRF tokens defend against forgery by uniquely identifying which user is making the request, allowing the server to authenticate the user based on the token alone

○ True ● False

Solution: The CSRF token does not authenticate the user. Instead, it verifies that the request originated from a page the server itself rendered

Q1.4 (1 point) Setting `HttpOnly=true` prevent CSRF attacks?

○ True ● False

Solution: Setting `HttpOnly=true` only prevents JavaScript from reading the cookie, but it can still be attached to requests.

This content is protected and may not be shared, uploaded, or distributed.

Q2 *Boogle* **(4 points)**

Boogle is a social networking website that's looking into expanding into other domains. Namely, they recently started a map service to try their hand at fusing that with social media. The URL for the main website is https://www.boogle.com, and they want to host the map service at https://maps.boogle.com.

Q2.1 (1 point) For each of the following webpages, determine whether the webpage has the same origin as http://boogle.com/index.html, and provide a brief justification.

i. https://boogle.com/index.html
ii. http://maps.boogle.com
iii. http://boogle.com/home.html
iv. http://maps.boogle.com:8080

Solution:

i. False. https://boogle.com/index.html and http://boogle.com/index.html do not have the same origin, since their protocols (https) and (http) are different.
ii. False. http://maps.boogle.com and http://boogle.com/index.html do not have the same origin, since their domains (maps.boogle.com) and (boogle.com) are different. The same-origin policy performs string matching on the protocol, domain, and port.
iii. True. The paths are not checked in the same-origin policy.
iv. False. http://maps.boogle.com:8080 and http://boogle.com/index.html do not have the same origin, because their ports (8080) and (80) are different. Note that if the port is not specified, the port defaults to 80 for http and 443 for https.

Q2.2 (1 point) Describe how to make a cookie that will be sent to only Boogle's map website and its subdomains.

Solution: Set the domain parameter of the cookie to .maps.boogle.com

Q2.3 (1 point) Boogle wants to be able to host websites for users on their servers. They decide to host each user's website at https://[username].boogle.com. Why might this not be a good idea?

Solution: A malicious user could set cookies that would be sent to other users' sites as well as the entire .boogle.com domain. Also, any cookies meant for boogle.com will go to the malicious user.

Q2.4 (1 point) Propose an alternate scheme so that Boogle can still host other users websites with less risk, and explain why this scheme is better.

Note: It's okay if the users' sites interfere with each other, as long as they can't affect official Boogle websites.

Solution: Boogle should create a new domain exclusively for user hosted content, like https://[username].boogleusercontent.com. This way, user sites cannot set cookies that will affect all boogle domains due to the cookie setting policy. This is known as a cookie tossing attack, and is one of the reasons why github hosts user sites on github.io instead of github.com (see https://blog.github.com/2013-04-09-yummy-cookies-across-domains/).

This content is protected and may not be shared, uploaded, or distributed.

Q3 *Web: Unscramble* **(7 points)**

`www.evanbook.com` is a website where users can submit and view posts. EvanBot is a user of this website, who is initially not logged in. Mallory is an on-path attacker between EvanBot and this website, and Mallory controls `www.mallory.com`.

- A user can load `www.evanbook.com/home` to see posts made by all users. (This behavior is the same whether the user is logged in or logged out.)
- A user can log in by making a POST request to `www.evanbook.com/login`, with their username and password (e.g. `"alice,password123"`) in the contents. If the username and password are correct, the HTTP response contains a session token cookie.
- A user who is logged in can load `www.evanbook.com/home?msg=X` to display all the posts, along with an additional message `X` at the top of the page.
- A user who is logged in can follow another user by making a GET request to `www.evanbook.com/follow?user=X`, replacing `X` with the username to follow.

In each subpart, provide a sequence of events (choosing from the list below) to execute the given attack. If you choose an event with a placeholder `X`, write the value you would insert into the placeholder.

A. EvanBot loads `www.evanbook.com/home`.

B. EvanBot loads `www.evanbook.com/home?msg=X`.

C. EvanBot makes a POST request with the correct username and password.

D. Mallory makes a post with contents `X`.

E. Mallory makes `www.mallory.com` send back `X`.

F. Mallory reads the HTTP request sent from EvanBot to `www.evanbook.com`.

G. Mallory reads the HTTP response sent from `www.evanbook.com` to EvanBot.

Write one event per row. You don't have to use all rows provided, but you may not use extra rows.

On each row: In the left box, write the letter (A to G) of the event. In the right box, if the event has a placeholder `X`, write the value you would use in the placeholder. If the event does not have a placeholder, leave the right box blank.

Example attack: Make EvanBot see the post "Mallory says hi."

Example answer:

D. Mallory makes a post with contents "Mallory says hi."

A. Then, EvanBot loads `www.evanbook.com/home`.

D	Mallory says hi
A	

This content is protected and may not be shared, uploaded, or distributed.

Q3.1 (2 points) For this subpart, assume all requests are sent over HTTP (not HTTPS), and the session token cookie has attributes `Secure=false` and `HttpOnly=true`.

Attack: Learn the value of EvanBot's session token.

Solution:

C. EvanBot makes a POST request with the correct username and password.

G. Mallory reads the HTTP response sent by `evanbook`.

The HTTP response contains the value of the session token cookie. Mallory is an on-path attacker, and the response is sent over HTTP (unencrypted), so Mallory can learn the value of the session token by reading this response.

Q3.2 (2 points) Attack: Make EvanBot log in as user `mallory` (who has password `161`).

From this subpart onwards, you may use the `post(url)` JavaScript function to send POST requests.

Solution:

D. Mallory makes a post with `<script>post("www.evanbook.com/login", "mallory,161"</script>`.

A. EvanBot loads `www.evanbook.com/home`.

This content is protected and may not be shared, uploaded, or distributed.

Q3.3 (3 points) Attack: Make EvanBot follow Mallory.

Solution:

D. Mallory makes a post with `<img src="www.evanbook.com/follow?user=mallory">`.

C. EvanBot makes a POST request with the correct username and password.

A. EvanBot loads `www.evanbook.com/home`.

This content is protected and may not be shared, uploaded, or distributed.