CS161 Fall 2025

Introduction to Computer Security

Discussion 7

Q1 Boogle (5 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.

websit	e is https://www.boogle.com , and they want to host the map service at https://maps.boogle.com .
	1 point) For each of the following webpages, determine whether the webpage has the same origin s http://boogle.com/index.html , and provide a brief justification.
	i. https://boogle.com/index.html
	ii. http://maps.boogle.com
ii	i. http://boogle.com/home.html
iv	v. http://maps.boogle.com:8080
-	1 point) Describe how to make a cookie that will be sent to only Boogle's map website and its ubdomains.
-	1 point) How can Boogle ensure that cookies are only transmitted encrypted so eavesdroppers on he network can't trivially learn the contents of the cookies?

Q1.4	(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?
Q1.5	(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 is okay if the user sites interfere with each other, as long as they cannot affect official Boogle websites.

(Question 1 continued...)

Q2 Session Fixation (2 points)

A *session cookie* is used by most websites in order to manage user logins. When the user logs in, the server sends a randomly-generated session cookie to the user's browser. The server also stores the cookie value in a database along with the corresponding username.

The user's browser sends the session cookie to the server whenever the user loads any page on the site. The server then looks the session cookie up in the database and retrieves the corresponding username. Using this, the server can know which user is logged in.

Some web application frameworks allow cookies to be set by the URL. For example, visiting the URL

http://foobar.edu/page.html?sessionid=42

will result in the server setting the **sessionid** cookie to the value "42".

Q2.1	(1 point) Can you spot an attack on this scheme?
	(1 point) Suppose the problem you spotted has been fixed as follows: foobar.edu now establishes new sessions with session IDs based on a hash of the tuple (username, time of connection) Is this secure? If not, what would be a better approach?

In a CSRF attack, a malicious user is able to take action on behalf of the victim. Consider the following example. Mallory posts the following in a comment on a chat forum:

Of course, Patsy-Bank won't let just anyone request a transaction on behalf of any given account name. Users first need to authenticate with a password. However, once a user has authenticated, Patsy-Bank associates their session ID with an authenticated session state.

Q3.1	(1 point) Explain what could happen when Alice visits the chat forum and views Mallory's comment
Q3.2	(1 point) Patsy-Bank decides to check that the Referer header contains patsy-bank.com. Will the attack above work? Why or why not?
Q3.3	(1 point) Describe one way Mallory can modify her attack to always get around this check.
Q3.4	(1 point) Recall that the Referer header provides the full URL. HTTP additionally offers an Origin header which acts the same as the Referer but only includes the website domain, not the entire URL. Why might the Origin header be preferred?
Q3.5	(1 point) Almost all browsers support an additional cookie field SameSite. When SameSite=strict, the browser will only send the cookie if the requested domain and origin domain correspond to the cookie's domain. Which CSRF attacks will this stop? Which ones won't stop? Give one big drawback of setting SameSite=strict.