Weaver Spring 2021

CS 161 Computer Security

Discussion 8

CSRF and XSS

Question 1 Cross-site not scripting

(

Consider a simple web messaging service. You receive messages from other users. The page shows all messages sent to you. Its HTML looks like this:

Mallory: Do you have time for a conference call? Steam: Your account verification code is 86423

Mallory: Where are you? This is important!!!

Steam: Thank you for your purchase

The user is off buying video games from Steam, while Mallory is trying to get ahold of them.

Users can include **arbitrary HTML code** messages and it will be concatenated into the page, **unsanitized**. Sounds crazy, doesn't it? However, they have a magical technique that prevents *any* JavaScript code from running. Period.

Discuss what an attacker could do to snoop on another user's messages. What specially crafted messages could Mallory have sent to steal this user's account verification code?

Question 2 Cross-Site Request Forgery (CSRF) () 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. (a) Explain what could happen when Alice visits the chat forum and views Mallory's comment. (b) Patsy-Bank decides to check that the Referer header contains patsy-bank.com. Will the attack above work? Why or why not? (c) Describe one way Mallory can modify her attack to always get around this check (d) 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?

won't it stop? Give one big drawback of setting SameSite=strict.

(e) 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

Question 3 CSRF++

()

Patsy-Bank learned about the CSRF flaw on their site described above. They hired a security consultant who helped them fix it by adding a random CSRF token to the sensitive /transfer request. A valid request now looks like:

https://patsy-bank.com/transfer?to=bob&amount=10&token=<random>

The CSRF token is chosen randomly, separately for each user.

Not one to give up easily, Mallory starts looking at the welcome page. She loads the following URL in her browser:

https://patsy-bank.com/welcome?name=<script>alert("Jackpot!");</script>

When this page loaded, Mallory saw an alert pop up that says "Jackpot!". She smiles, knowing she can now force other bank customers to send her money.

- (a) What kind of attack is the welcome page vulnerable to? Provide the name of the category of attack.
- (b) Mallory plans to use this vulnerability to bypass the CSRF token defense. She'll replace the alert("Jackpot!"); with some carefully chosen JavaScript. What should her JavaScript do?
- (c) patsy-bank.com sets SameSite=strict for all of its cookies. Does this stop the attack from part (b)? Assume the welcome page does not require a user to be logged in.
- (d) Mallory wants to attack Bob, a customer of Patsy-Bank. Name one way that Mallory could try to get Bob to click on a link she constructed.