

CSRF and XSS

Question 1 *Cross-site not scripting* (0)

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?

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

