Eric Koerlin
Unit 08

1. (2) HTTP is a stateless protocol - what does that mean?

   This means that HTTP by default knows nothing about the current state that you are in, it doesn't know if you've logged in, which user you are, or anything like that, it means that if you need to log in to get access to a page, then you must log in at each request since no state information is stored.

2. (2) What is a session id?

   A session id identifies a session as belonging to a user, which allows stateful interaction.

3. (4) Explain the relationship between cookies and sessions

   A cookie gives a place to store identifying information so that the right session can be rendered for the user.

4. (2) Name two guidelines for storing sessions

   Set sessions to expire

   Don't store any secrets in a session

5. (4) What is CSRF?

   CSRF stands for Cross-Site Request Forgery and is an attack where content is rendered on a page that sends a request to a page, potentially on a different domain, but one that the victim is logged in to, so that the request is validated with the victim's authentication cookie, which is provided by the browser by default. This allows the attacker to submit requests to do things like destroy accounts, or delete records.

6. (4) What steps does Rails take to protect the user from CSRF? Note: You might want to look at `application_controller.rb` and `application.html.erb` - after this reading you should understand more about some of the options in these files.

   The default Rails behavior is to generate a hidden, secret form field on a page that must be provided with any POST, PUT or DELETE request to enable the change to be made. This means that the page must be rendered in order to be able to make a request, and not just authenticated with a session token.

7. (2) Compare whitelisting to blacklisting. How does this relate to the params hash? (not explicitly mentioned in reading)

   Blacklisting says these finite limited things are not allowed, while whitelisting says, only these specific things are allowed. Whitelisting is more secure because there are a finite number of things it lets in, and an infinite number that it keeps out, whereas blacklisting has a finite number of things to block and an infinite number of things to allow. This means that someone more clever than you, or with more time on their hands can likely find a way to come up with something that isn't on the blacklist to do what they want. The params hash in Rails lays out specifically what sorts of things it will accept, and is an example of whitelisting.