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

This means that HTTP cannot keep track of the certain state of a particular user. Therefore, by using sessions, Rails can identify the current user, or what is currently in the shopping cart for a user. In a stateless protocol, "the user would have to identify, and probably authenticate, on every request." (http://guides.rubyonrails.org/security.html)

2. (2) What is a session id?

"The session id is a 32-byte long MD5 hash value." The session id is a hash value constructed from a random string based on the current time, a random number between 0 and 1, the process id number of the Ruby Interpreter, and a constant string. It is used by the browser and the server in order to keep track of the state for a user's session, in order to maintain some line of secure communication. It cannot feasibly be brute-forced, but it is not collision-resistant.

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

"The session id in the cookie identifies the session…the cookie serves as temporary authentication for the web application." The cookie is how the user authenticates themselves with each request. Therefore, if an attacker were to get a hold of a user's cookies, they could pretend to be that user in that session.

4. (2) Name two guidelines for storing sessions

When storing sessions, "*Do not store large objects in a session*." This is because large objects will quickly fill up your session storage space and if you choose to modify the structure of an object, this makes things easier to handle. Also, "*Critical data should not be stored in session."* because if a user were to clear their cookies or closes the browser, the critical data would be lost.

5. (4) What is CSRF?

Stands for cross-site request forgery, in which an attacker includes malicious code or a link in a page that accesses a web application that the user is believed to have authenticated. Given the session for the application has not timed out, the attacker can execute unauthorized commands. "It is important to notice that the actual crafted image or link doesn't necessarily have to be situated in the web application's domain."

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.

"To protect against all other forged requests, we introduce a *required security token* that our site knows but other sites don't know." These tokens are included in the requests and verify a request in the server. `protect_from_forgery with: :exception` can be found in the application controller, and thus rails will include a security token with requests.

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

Both whitelisting and blacklisting create lists which are compared to the input. A whitelist will contain items that are deemed correct or acceptable, while a blacklist will contain items that are invalid. The params hash comes from the users browser when they request a page. Using these methods, the params hash can be verified in order to allow or deny certain parameters from being acted upon.