Nicole Holden, Michael Rhodes, Michael Terranova

Activity 4:

1. This was done by a single individual while other work was being done. So the graph here would be vertical.
2. Travis tests
3. Sessions are only established after the user is authenticated and are generated internally at the time of authentication.
4. Since we are using RITs LDAP server and real student credentials, we never store or display credentials anywhere. Users authenticate over HTTPS only, via a Java microservice using the Spring framework with security settings configured and encryption enabled. The request is passed on the RITs LDAP server which authenticates the user for us. At no time are credentials stored for the user.
5. RITs LDAP server, and login services are rate limited, after so many unsuccessful login attempts users are unable to login to their account for some period of time. To prevent this feature from being abused, the Spring framework can be configured to add additional login limiters such as frequency, and quantity of attempts.
6. Username enumeration is prevented by providing the same message back to the user regardless of whether or not the username/credentials exist, and by ensuring the responses for valid and invalid credential searches take roughly the same amount of time (to prevent timing attacks).
7. A predictable sessionID could result in a session prediction attack, which could result in a user's session being hijacked by the attacker, leading to the attacker having control of the session without needing to authenticate.
8. There are many downsides to centralized authentication. Some being: if SSO goes down, all users lose all access to the sites content, stronger passwords would be necessary to prevent someone from gaining access to all the sites information, rather than just a small chunk, and all user's sessions may not expire at logout, but will require the user to entirely close their browser, if they forget this step, an attacker would have an easy way back into the website.

Activity 6:

1. Burndown chart
2. Travis CI
3. We would have prevented this with token based mitigation within our php script. We already validate sessionID and the source IP address as well.
4. We would limit the size of the file that can be uploaded, ensure it is not an EXE, and verify it through an online malware checking website.
5. We would prevent this by using the HttpOnly flag in our script, filtering output, and utilizing PHP's built in sanitization functions, such as "sanitize_email()"

Activity 7:

1. Burndown
2. Travis
3. Closing external ports and requiring authentication to access them

4. We could remove any dangerous skits or not allow the skit in general. If we go the latter route, we could display an error message of “Your Skit has banned character in it and is not allowed”.
5. We would only allow unicode characters from within a certain range, such as the english alphabet and all allowed characters within Skitter.
6. We would have a javascript function on the pages that allow skitting and if it is found the characters are greater than 140, not allow the skit. We could also double check the skit with a php script on our end to ensure they didn’t locally change the code to allow their skit to go through.
7. None, we didn't implement any. It probably is a problem.

Activity 9:

1. The amount of time a reply takes to receive could be interpreted to mean various quantities of replies and/or the presence of the user or replies in general. Timing attacks have been used to guess usernames (there was a discrepancy between a login attempt with a valid user and invalid user), and other potentially sensitive information in many attacks. Beyond that, usernames themselves aren’t a secret (both the username and display name are shown). If the username was a secret and only the display name was shown, and the skit replies were stored by username or userid, a bad guy could potentially use the timing attack of searching for replies to guess usernames by modifying requests.
2. More travis