# Activity 1: Setup GitHub

## Questions

1. What is the URL of your Github project?

<https://github.com/CSEC380-Group16/csec380-project>

1. How did you breakup your projects and what are the security ramifications?

We broke up our project into tasks based on the activity requirements. Security ramifications could result from a single team member being responsible for an implementation and overlooking the security of it. We will try to mitigate the chance of that happening by placing tasks in the “Review” column of the project for all team members to review, prior to it being moved to the “Done” column.

1. How did you choose to break down your Epic into various issues (tasks)?

We read over the project requirements for each activity and picked out tasks that seemed like incremental steps to accomplish the activity while maintaining substance.

1. How long did you assign each sprint to be?

Approximately 2 weeks to be flexible with other homework assignments for the class.

1. Did you deviate from the Agile methodology at all? If yes, what is your reasoning for this?

No we did not deviate from the agile methodology. We implemented issues/tasks using GitHub’s “Issues” tab, sprints with due dates under the “Milestones” tab, and broke down the project into each activity in the “Projects” tab with labels to associate each issue with the activity.

1. How do you ensure that after each issue/milestone that security has been verified? How would you identify such issues in an ideal environment?

We have created a column for each activity called “Review” to make sure that everything is compatible to prevent falsely marking it as “Done”.

# Activity 2: Generate Tests/Design (UML)

## Questions

1. What Web Application security mechanisms are involved in your topology? What security mechanisms would ideally be involved?

The main security mechanisms involved in our topology are part of the authentication system. Users are required to enter a username and password before using any part of the web app. The passwords will be stored as hashed values. Users sessions will expire after inactivity to mitigate the risks involved with session hijacking.

1. What testing framework did you choose and why?

We chose pytest because there is a lot of documentation on it to help us implement it.

# Activity 3: Authentication (PHP/Python/Ruby/Go/NodeJS)

## Questions

1. Provide a link to the test cases you generated for this activity.

Test cases can be found in the test directory labeled by activity.

1. How do you ensure that users that navigate to the protected pages cannot bypass authentication requirements?

Authenticated with a form submission instead of a separate page and uses sessions.

1. How do you protect against session fixation?

We used PHP session\_start.

1. How do you ensure that if your database gets stolen passwords aren’t exposed?

We hashed the passwords before entering them into the database.

1. How do you prevent password brute force?

We lockout a user’s session for 2 minutes after 6 failed attempts.

1. How do you prevent username enumeration?

The login page displays the same message if username or password are incorrect, so you don’t know which one is actually incorrect.

1. What happens if your sessionID is predictable, how do you prevent that?

PHP prevents it for us with their built-in session management. We regenerate a new session ID with every visit to the login page.

# Activity 4: The Content (PHP/Python/Ruby/Go/NodeJS)

## Questions

1. How do you prevent XSS is this step when displaying the username of the user who uploaded the video?

We decided to not allow users to register with usernames containing special characters. We use the PHP function htmlspecialchars() to handle the username, and when it is prepared by MYSQLi, it throws an error and does not add the user to the database, therefore not creating their account.

1. How do you ensure that users can’t delete videos that aren’t their own?

We check the database to ensure the video they entered is associated with their user ID.

# Activity 5: SQL Injection (Classic/Blind)

## Questions

1. How would you fix your code so that these issues were no longer present?
2. What are the limitations, if any that, of the SQL Injection issues you’ve included?

# Activity 6: SSRF

## Questions

1. How would you fix your code so that this issue is no longer present?

I would check that the entry in upload by URL is a valid URL that uses HTTP or HTTPS.

1. How does your test demonstrate SSRF as opposed to just accessing any old endpoint.

This test demonstrates that one user can copy another user’s video and claim it as their own. If there was a secret video that was made private on the server, that could be displayed as well.

# Activity 7: Command Injection

## Questions

1. How would you fix your code so that this issue is no longer present?

To fix this issue, I would remove in this case the shell\_exec() function in favor of referring to the database for a user’s videos. In any case, I would avoid ever using the commands listed by OWASP for testing command injection: <https://www.owasp.org/index.php/Testing_for_Command_Injection_(OTG-INPVAL-013)>