Wyatt Tauber
Shannon McHale
Connor Shade
Mike Lanzafame

Activity 1 Questions

https://github.com/wwt9829/CSEC-380-Project/wiki/Activity-1-Questions

- **What is the URL of your Github project?**

  https://github.com/wwt9829/CSEC-380-Project

- **How did you breakup your projects and what are the security ramifications?**

  Each activity in the writeup is one GitHub project. The writeup was created in a logical order (document program, design program, implement vulnerabilities in program) by an experienced web application developer, and we see no need to deviate from this plan currently.

  - *Following are several security ramifications that we will keep in mind when creating the rest of our project.*
  - **User authentication** - Authenticating users in a secure manner is very important. If not done properly, our web application could store passwords in plaintext, be vulnerable to cookie or session hijacking (if cookies are used), or implement client-side authentication.
  - **File processing** - Ensuring that only videos are uploaded is another concern. An attacker could possibly format malicious code with the header or file extension used by a video and upload it to our application to be processed. If we are unable to detect and prevent this, our server could be put under an attacker's control.
  - **Web server security** - We will also need to configure the web server securely, as users on the internet will have access to it. This includes checking for problems such as file inclusion vulnerabilities, unsafe evaluations, and ensuring our web app is using the latest version of the web server.
  - **User input** - Users will interact with our application in both expected and unexpected ways. Our project will need to properly handle user input and ensure that a malicious user cannot send our web server commands via inputs.
  - **GitHub** - Putting our program design documentation and code on GitHub will make it easier for an attacker to find and exploit vulnerabilities, but it

will also make it easier for us to identify vulnerabilities and collaborate with others to fix them.

- **Docker containers** - By implementing our entire infrastructure using Docker containers, several risks have been introduced, such as the susceptibility to container breakout attacks, kernel exploits, or denial-of-service attacks. However, it is possible to configure Docker in a way that successfully mitigates the risks that come along with using Docker.

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

  After reading the activity user story and goal, we created an issue for each actionable item or deliverable we thought necessary. This included making issues for items such as the UML diagrams, setting up continuous integration, and writing unit tests. Each team member was then assigned two to three tasks depending on their availability.

- **How long did you assign each sprint to be?**

  Activities 1 and 2 were assigned to the same sprint as they both cover the configuration of services and the creation of documentation. This sprint (sprint #1) was assigned for a two-week period. Activities 3 and 4 were each assigned a sprint twice as long (4 weeks) to accommodate for the creation of most of the back and front ends of the website, as well as to allow adequate time for troubleshooting and design/security fixes if necessary. The implementations for each vulnerability were each assigned a 1-week sprint length as they are believed to be simpler to implement than the website itself.

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

  Typically Agile sprint lengths are the same (i.e. 2 weeks for each activity). However, we have a fixed size team and limited resources, so providing adequate time to implement the most complicated parts of the project were a priority. We believe the varying time length for each sprint will give us the opportunity to understand our abilities and development rate; we can then adjust the sprint length accordingly.

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

In addition to writing unit tests to check for desired operations and check against undesired operations (such as vulnerabilities introduced inadvertently), we will use Codacy to provide additional code coverage, security, and style tests.