Hatcher Blair

CS-405 Secure Coding

12/15/2023

Module 8 Journal

Security is something that we all know is important yet is often forgotten about or left until the end of development. This often happens because of time constraints, cost, or people deciding it is not that important. However, when your product is on the news because of a security breach, it becomes a big problem that is expensive to remediate. One of the strategies to prevent this from happening is implementing a secure coding standard. This standard gives the developers clear and detailed guidelines for how the product handles security. It also serves as a reference for the team while working on the product. The development of this policy needs to start from the beginning of development and is a living document. Starting at the beginning of development allows the groundwork for security to be implemented at the core of the product. This means that the rest of the product will be built on top of the secure foundation and that security will always be a consideration for the developers working on the project.

Remediation is very expensive and comes after a wave of bad pr but, mitigation is also very expensive. However, mitigation is almost always cheaper and does not come with a wave of bad pr from a data breach. It is important to assess what mitigation techniques you should be implementing, in some instances security might not be that important. For example, if you run a website that publishes recipes for others to use. The risk associated with someone attacking your website is negligible. You are not going to be a target because you don’t store user data and the information that you publish is not very valuable. This means that your security does not need to be the best in the world. It is important to not only address security but also to assess the cost of implementation, and the risk of an attack occurring.

One strategy that is commonly used and should be used for all products is zero trust. Zero trust is trusting absolutely nobody. This means that all users start out with zero permissions and over time they can get more permissions, usually in the form of a role-based system. Going back to the example of the recipe website, you wouldn’t want any user to be able to upload a recipe to your site. This means that all users that enter the website would have no ability to do anything other than view the site, they are not trusted. However, you probably want all users to be able to make comments on the recipe so you can get ideas on how to improve your recipes or see what people want recipes for. So, you have people make an account and now they can post comments on recipes. The users provided some basic information about themselves and now that you have a little more trust in them, they can make comments on the recipes. Now you add another person to your team. This person is there to provide more recipes and they need to be able to upload recipes to the website as well as make changes to existing recipes. However, you don’t want the new employee to be able to make changes to the layout of the website or take the website down. To solve this, you make another account type for employees that allows them to edit recipes and upload new ones. They cannot make any significant changes to the website because they do not have enough trust yet. This is essentially how zero trust functions.