Podcast 066- AN INTERVIEW WITH SHARI LAWRENCE PFLEEGER

This podcast focuses on the need for strong requirements in secure software. There are two different types of software that are focused on that both play a part in quality software. Safety critical systems are needed to protect people from harm, and security critical software defends against other people’s ability to get inside of the system.

No matter how much we advance the technology, people are always thinking of new ways to get around and into the system. The adversaries are always thinking of new ways we may not have thought of defending to penetrate a system. One idea to improve this aspect of development would be a meeting between security teams and criminologist to help understand how intention works and how potential adversaries think. Security does not always just deal with the software or technology and how to prevent intrusion but also includes the different ways people may decide to use the software.

A lot of security requirements come out of having the correct system requirements and scope. Architectural and requirements analysis plays an even bigger role in security than code reviews or the code itself. Often security is left out of requirements gathering. It is a learned practice and unless specifically developed it can often be flawed or overlooked. Using tools to help find requirements does not really help, instead having a team focus on requirements and the additional time spent thinking about them is what really helps develop complete requirements. Developers need to broaden perspective and make sure the software they are developing is secure.

Nobody explains the negative aspects of bad code and the damage it can cause until its too late. Compared to say mechanical engineering where they stress system failure and the real life risks to people, yet software classes do not stress the real world risks that bad code can
have and the damage it can cause.

However, it is often hard to measure security metrics. People are over booked and don't have the funds to add more data collection to a project. Often metrics measure too much and do not get results that are beneficial. Instead, a team should measure a few specific aspects of a system and use the results to increase profits and quality of code. If is often a good practice to develop standards and processes that allow better quality code development.