What is your role in solving security concerns as a developer?

The role of the developer is to anticipate where attacks may come from by identifying what information in the application is valuable to the attacker. From this, a developer can see what portions of code interact with this information database and ensure that security is robust enough to disallow negligent access based on known vulnerabilities.

What might solving security concerns as a developer involve?

Solving security concerns as a developer involves an understanding of known vulnerabilities through the use of static testing and research in contrast to how the code is intended to behave. According to the DevSecOps methology, this is a continuous process built in-line with regular DevOps.

Where does security fall within the software stack and development life cycle?

According to DevSecOps, security will fully integrate into the complete software development life cycle (SDLC). This is still somewhat new to the world of development and cooperation and buy-in are needed for the success of implementation. Continuous Security is added to the Continuous Integration and Continuous Delivery (CI/CD) model. Simply put, each iteration of code is rigorously tested against known attacking attempts and passed before proceeding onto further development stages. Security is considered at every stage of the development process to minimize vulnerabilities.

How might you add security measures to transform a DevOps pipeline into a DevSecOps pipeline?

Transforming a DevOps pipeline into a DevSecOps pipeline involves integrating security through all development and delivery processes. You must first sell this integration and personally own it if you want others to adopt this style. Each member on the DevOps side need to be provided with security training to understand the critical components of why complete security is necessary. As per Mr. Jaganathan, things like integrating automated security scanning tools and continuous static application security testing/dynamic application security testing, regular peer code reviews, and implementation of container security scanning for containerized applications are a few great steps in that transformation.

The article suggests creating and following a plan to secure the entire DevOps life cycle. What is included in the suggested plan and would you recommend following it?

The included plan recommends seven stages of development; plan, code, build, test, release, deploy, and operate. A brief synopsis of this plan includes starting with a rapid risk assessment to evaluate threat models and quantify risks with in the plan step. Within the code step, this is where you want to apply secure coding practices and integrating automatic testing such as static application security testing and conducting peer code reviews. For build, you want to check software modules into the source code repository and segregate the repository by production vs. non-production environments to ensure untested code does not leak into the production causing vulnerabilities down the line. The test step is one of the major differences between the two models. Even though testing is integrated throughout, here is a more extensive testing dividing into teams, one for offensive and the other for defending against these offensive attacks. Next is the deploy stage, involving processes, tools, and technologies for releasing software modules into lower and live environments based on defined schedules and policies. Finally, the last step, operate, manage the live-run environment. Here, the security team is actively involved in continuous monitoring and analysis.

I believe this is a very acceptable plan to follow. I would recommend this process as it is a sound way to integrate the security into DevOps. This process even recommends tailoring to whatever organization you plan to adopt it to. As this is a new way of thinking, it is always a good idea to start with a baseline plan, and especially one that encourages fabrication.