Eric Williams-Phillips

Bellevue University | CSD 380 | Module 12

Sue Sampson

**Compliance in Production**

In both of the case studies “Providing Compliance in Regulated Environments” and “Relying on Production Telemetry for ATM Systems”, the author discusses the overreliance on traditional code reviews as the main method of finding vulnerabilities or security breaches in software. According to Bill Shinn, a principal security solutions architect at Amazon Web Services, auditors have been taught to search for problems post-development, which usually leads to practicality and reliability issues. He made a great point in observation of the relationship between auditors and developers:

*“…how many auditors can read code and how many developers have read NIST 800-37 or the Gramm-Leach-Bliley Act?”*

The point of DevOps is to bridge that gap in knowledge to prevent any potential threats from being able to slip through. As the quote says, each role has their responsibilities and their own training, but they still rely on one another to ensure that compliance is still being followed. The conflict is determining what the most efficient way of bridging that gap.

Shinn was the primary source used in the case study “Proving Compliance in Regulated Environments”. The idea behind this case study was to prove that customers still comply with laws and regulations if they merge auditing tasks into the software development lifecycle. Not only do they abide by regulations, but this practice is generally better than requesting a sample from an already large and established infrastructure. Instead, Shinn’s teams work alongside auditors to do incremental security checks to ensure that auditors are satisfied during production and whenever they need it. This is due to telemetry systems like Splunk and Kabana that are sent data automatically so that auditors can retrieve it without waiting and can verify that not only finalized, but in-development codebases are following protocol. These systems are designed to collect and process mass amounts of data that can be parsed for specific information. Additionally, the goal is to present the data in a clear and understandable way for auditors rather than “screenshots and CSV files filled with configuration settings and logs”.

In “Relying on Production Telemetry for ATM Systems” the head of a large US-based financial organization’s DevOps team discusses an incident in which production monitoring controls have been supremely beneficial for the team over post-production auditing. The incident involved a developer injecting a backdoor into the codebase that allowed them to manipulate the ATM maintenance modes during certain times and take cash out. Through the use of production telemetry, or transparent automated production data, the team was able to discern the fraud before any scheduled security audit. The idea is to take use of telemetry as a tool for auditing instead of separating the duties and potentially missing vulnerabilities.

Both case studies point toward the overreliance of code reviews and sampling to detect vulnerabilities and security breaches and suggest teams share the responsibility to cover all bases. Production telemetry reduces the amount of preparation required for security audits and is much more meticulous than a sample of an already established system. Shinn’s iterative approach supports a thorough review of code in-development and because it is done during the lifecycle, it is much more scalable and reliable.

Works Cited:

Kim, G., Debois, P., Willis, J., & Humble, J. (2015). *The DevOps Handbook: How to Create World-Class Speed, Reliability, and Security in Technology Organizations*. It Revolution Press.