**Module 12.2 Assignment**

CSD 380 – DevOps

Darren Osier

Jeremy Ginter

7/27/2024

In the case study “Providing Compliance in Regulated Environments” Bill Shinn, a principal security solutions architect for Amazon Web Services, explains some of the common problems with ensuring large enterprise customers are able to comply with laws and regulations (Kim et al., 2016). Shinn explains that those who audit these regulations are trained to do so in ways that aren’t suited to common DevOps practices. One example he gave is with large amounts of production servers. An auditor would likely want to see a sample of these servers. While this would be feasible in a physical environment, auto-scaling makes it so servers appear and disappear depending on usage, making it impossible to sample (Kim et al., 2016). Deployment pipelines also create an issue, since separate groups are responsible for writing and deploying the code.

In order to solve these issues, Shinn works with the auditors to find different methods to show auditors evidence of operational controls (Kim et al., 2016). One method of doing this is to work with the auditors while the development team is creating the control design. This allows auditors to get the information they need on demand during the development process. One way this is done is through telemetry systems, which receives all data and can be accessed by auditors to obtain evidence whenever it is needed (Kim et al., 2016). The DevOps Audit Defense Toolkit is another useful tool for organizations, allowing them to see an example of how to fulfill requirements that will likely be audited. The DevOps Audit Defense Toolkit is in essence an example of an audit of a fictional company. This can be used to show top risks, the control environment, and management practices that prove controls are in place and are working (Kim et al., 2016).

In the case study “Relying on Production Telemetry for ATM Systems, the head of the DevOps initiative for the banking property explains how Information Security, auditors, and regulators put too much stake in code reviews as a method to detect fraud (Kim et al., 2016). She claims that rather than relying solely on this, production monitoring controls, code reviews, and approvals should be integrated in order to better prevent fraud from occurring.

One instance of this being a problem is when a developer planted a backdoor in the code for the company’s ATM machines (Kim et al., 2016). This backdoor allowed the developer to put these machines into a maintenance mode during certain times. This gave them full access to take the cash out of the machines. She explains that while the fraud was able to be detected quickly, it was not discovered through code review, and that similar issues are not easily detectable through this method. What caused this issue to be discovered was someone noticing the unscheduled maintenance and bringing up the issue at a regular operations meeting (Kim et al., 2016). This is a perfect example of how production telemetry can be useful in finding issues and fraud that code review would be unable or unlikely to catch. I believe this makes reviewing production telemetry extremely important for the security of applications and software, and makes consistent review of production telemetry a great way to detect fraud before it becomes a large issue.

**References**

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2016). *DevOps Handbook*. IT Revolution Press.