Joseph Ayo

CSD 380

7/26/25

Bellevue University

Case Study Review – The Devops Handbook

In Chapter 23 of *The DevOps Handbook (2nd Edition)*, two case studies are presented to us; “Proving Compliance in Regulated Environments” and “Relying on Production Telemetry for ATM Systems”. These case studies focus on the real-world challenges that organizations face when applying DevOps practices in highly regulated and security intensive industries, like banking. Both case studies emphasize the shift from traditional compliance and security practices to more specialized ones, to fit the needs of the job better. In this paper, we will discuss the lessons within each case study and talk about the benefits that come from the adjustments developers made in both studies.

In the first case study, Bill Shinn, a principal security architect at Amazon Web Services, describes the challenge of proving compliance in cloud environments using DevOps methods. Traditional audit techniques that Bill mentions, like requesting screenshots and CSV logs, are simply not good enough for dynamic infrastructures where servers appear and disappear frequently due to auto-scaling and infrastructure-as-code. Auditors, trained in older methods, struggle to interpret the newer forms of evidence DevOps generates. Shinn points out the mismatch between what auditors expect (e.g., physical server samples) and what modern cloud-based environments can provide (e.g., immutable infrastructure, automated deployments). To resolve this, Shinn’s team collaborates with the auditors during the control design phase to align what constitutes valid evidence and how it will be collected.

The second case study, “Relying on Production Telemetry for ATM Systems,” illustrates a practical security incident in a large U.S. financial organization. Mary Smith, a leader in the company’s DevOps transformation, recounts how a developer had embedded a backdoor in ATM software, allowing unauthorized cash withdrawals by putting machines into maintenance mode. Despite the organization’s use of code reviews and change approval processes, the fraud was not caught during development. Instead, it was identified through real-time monitoring when someone noticed unusual patterns during a routine operations review meeting. ATMs in one city were going into maintenance mode at unscheduled times, which was an anomaly that was only visible due to production telemetry.

This case study reveals something important to us: that fraud and system integrity violations are often not caught in code but are instead seen through behavior. Code reviews, while valuable, can miss subtle or deliberately obscured vulnerabilities, like the ATM back door. Production telemetry, on the other hand, provides continuous observation of the system’s behavior, allowing teams to detect anomalies in real time. By monitoring system behavior in addition to the standard preventive measures, the organization was able to quickly detect and resolve the issue before a formal cash audit even took place.

Together, these case studies emphasize the importance of shifting compliance and security mindsets from static, documentation-heavy processes to more dynamic, data-driven ones. Both Shinn and Smith argue for embedding compliance into the DevOps lifecycle through close collaboration and automated data observation. Rather than treating compliance as a roadblock, these stories show how integrating audit evidence generation into CI/CD pipelines and leveraging telemetry not only satisfies the cumbersome requirements but also enhances a program's security.

In conclusion, these two case studies show us that modern compliance and security in DevOps environments have forced us to rethink how we audit our programs, to further enhance software security. The Amazon case demonstrates how to automate and standardize audit evidence using telemetry and version-controlled infrastructure. The ATM case illustrates the power of production telemetry to detect anomalies that slip through traditional code review. I believe that these lessons teach us that there is always room for innovation, and that code standards exist as a baseline, not the final line, towards building the most secure product possible.