Paul Romer

CSD380

Module 12 Assignment

Compliance

Chapter 23 of The DevOps Handbook focuses on answering the question “How do we stay safe while development pace is quick and agile?

Providing Compliance in Regulated Environments

Bill Shinn, a principal security architect at Amazon Web Services, spends his days helping highly regulated customers prove to external auditors that their cloud environments meet and exceed regulatory requirements. Many regulators still expect server screenshots and CSVs with logs, which was common early in the software development industry when changes took weeks, months, or even years. That legacy way of auditing software doesn’t work well, especially as DevOps principles are adopted by the enterprise. When software changes are pushed from each sprint to many times per day, automated CI/CD pipelines, and short-lived infrastructure that auto-scales as needed, it changes how these systems should be automated. Shinn explains that because of this, changes to how audits are done are needed. Their teams bring auditors into each sprint. With the auditors, they determine what is needed in terms of audit evidence and build dashboards and systems to ensure they have the information that they need on demand. This lowers the probability of audit and security issues.

Relying on Production Telemetry for ATM systems

Mary Smith (a pseudonym) leads DevOps for the consumer-banking arm of a large U.S. financial firm. Years ago, a rogue developer hid a back door in the code deployed to thousands of ATMs. By toggling machines into “maintenance” at odd hours, the developer could withdraw cash at will. Classic safeguards were present, like separation of duties, formal code reviews, a change-approval workflow. None of them spotted the exploit. What saved the day was routine ops review. During a weekly dashboard check someone noticed a weird cluster of unscheduled maintenance events in one city. That anomaly triggered an investigation, the fraud was verified, and the hole was patched before the next physical cash audit would have caught the shortfall.

Both stories underline a simple DevOps mantra: “If it isn’t in telemetry, it didn’t happen.” Shinn’s team demands telemetry as proof to outsiders; Smith’s team uses telemetry as security. Both teams benefit from the same auditability rules:

- Feed every meaningful event, build, deployment, permission change, and runtime state, into a searchable, time-stamped log store.
- Give every stakeholder (auditor, engineer, security analyst) direct, read-only access.
- Treat the resulting stream as a living specification of how the system works and whether it is working right now.

By embedding compliance and security feedback loops directly into daily engineering work, teams gain speed and safety. The stories also carry a caution. Controls that live outside the deployment pipeline (manual screenshots, once-a-year reviews) don't work in fast-moving environments. They can't keep up with the pace of development. Controls that work with the code, like automated tests and observable runtime behavior, get more useful with every commit. The goal is fast and transparent feedback for all stakeholders. With fast and transparent feedback, audits become easier, fraud becomes harder, problem resolution is easier, and everyone is happy.