Faye Van Roekel

Assignment 9.3 Compliance

February 28, 2020

This paper covers the case study in our textbook that talks about how large companies prove that they are in compliance within regulated environments with using Telemetry. “In pretty simple terms telemetry is the process of recording the behavior of your systems” (Gutta & Angara, 2016).

Bill Shinn, Principal Securities Architect at Amazon Web Services was the person who was tasked with the responsibilities to comply with all relevant laws and regulations. Over the year, Shinn spent time with over one- thousand enterprise customers who have publicly referenced their use of public clouds in highly regulated environments. He found that the auditors that were employed, have been trained in methods that are not very suitable for DevOps work patterns. It is very difficult to audit the infrastructure, when auto-scaling makes servers appear and disappear. When this happens, there is not a good way to send the auditor a sample, especially within the deployment pipeline, which is different from the traditional software development process. One group of employees writes the code and the other group deploys that code into production. Shinn’s goal was to create some alternative methods of presenting the data which clearly shows auditors that their controls are operating and are working effectively.

Shinn had created some dedicated teams to help bridge the gap with auditors in the control design process. These teams use an interactive approach by assigning a single control for each sprint. This determines what is required to present as audit evidence. This also helps ensure that auditors get the right information that is needed when the service is in production, which functions entirely on demand. Shinn ended up sending all data to the telemetry systems and the auditors then could get what they needed, and it was completely self-serviced. There is now no need to request data samples, instead they log into the system and retrieve the audit evidence they need. With the modern audit logging, chat rooms and deployment pipelines, it has resulted in unprecedented visibility and transparency, with what is happening in production. This is especially true compared to how operations used to be conducted with lower probability of errors and security flaws being introduced.

Shinn explains in the case study that HIPPA can be tricky to figure out what is required from the actual regulations. HIPPA requirements from an information security standpoint can be found in forty-five CFR Part 160 legislation, Subparts A and C of part 164. When you find technical safeguards and audit controls, you know you are in the right place. There you will find what is required is that we need to determine activities that will be tracked and audited relevant to Patient Healthcare Information, document and implement those controls, select tools, and then finally review and capture the appropriate information.

The DevOps Audit Defense Toolkit is a document that describes how controls could be designed in a deployment pipeline to mitigate the stated risks, and provides examples of control attestations and control artifacts to demonstrate control effectiveness. This document was intended to be general to all control objectives and can be extremely helpful getting developers started down the right path. Compliance in regulated environments is a large undertaking for any company and something that should done in an efficient and proper way. Customers and companies have vital and confidential information which must be protected at all times.

**References:**

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2017). The DevOps Handbook: how to create world-class agility, reliability, and security in technology organizations. Portland, OR: IT Revolution Press

Gutta, S., & Angara, J. (2016). Devops Product Line Engineering (Dple): Where Devops Meets Software Product Lines. *PONTE International Scientific Researchs Journal*, *72*(11). doi: 10.21506/j.ponte.2016.11.35

Palermo, J. (2019). Operating and Monitoring the Release. *.NET DevOps for Azure*, 247–264. doi: 10.1007/978-1-4842-5343-4\_10