Unit 11: Future trends in Secure Software Development

Debate: Microservices and Microkernels

Task:

Read Appendix A: the Tanenbaum-Torvalds debate in DiBona & Ockman (1999) then read Fritzsch et al (2019).

The forum has a message that says: "Torvalds has been proven wrong and it only took nearly thirty years. Microservices and microkernels are the future. "

On the forum post a message either agreeing or disagreeing with the above and give a justification (ideally with an academic reference) supporting your view.

Initial Message:

The assertion that "Torvalds has been proven wrong and it only took nearly 30 years. Microservices and microkernels are the way of the future" is an oversimplistic view of the trade-off between monolithic and microkernel designs and between monolithic architectures and microservices. Although microservices and microkernels have begun to make inroads in modern computing, nothing should throw into question the ongoing applicability of monolithic systems, for instance, in specific scenarios.

The 1992 Tanenbaum-Torvalds argument exemplifies the conceptual contradiction between microkernels and monolithic kernels. Tanenbaum argued that microkernels represent a forward-looking design paradigm that emphasizes modularity, portability, and fault isolation. He argued that microkernels offer cleaner abstractions and better maintainability because of the isolation of components in user-space processes

(DiBona & Ockman, 1999). In contrast, Torvalds stressed the pragmatic benefits of monolithic kernels, focusing on their ease of development and improved performance given the technical limitations of the early 1990s. Torvalds acknowledged that while microkernels were theoretically appealing, their implementation complexity and performance overhead made monolithic systems a more pragmatic choice (DiBona & Ockman, 1999).

Over the past few years, the microservices architectural paradigm has been heavily used, especially for cloud-centered and large-scale systems. Fritzsch et al. (2019) highlight the advantages of microservices, including improved scalability, fault tolerance, and alignment with modern DevOps practices. However, decomposing monolithic applications to perform tasks into microservices is labor-intensive and technically difficult. It often requires careful consideration of service granularity and may introduce new complexities, such as dependency management and increased latency owing to distributed communication. Fritzsch et al. (2019) caution that poorly executed migrations could result in hybrid systems that fail to leverage the benefits of either architectural style effectively.

Notwithstanding the appeal of microkernels and microservices to specific application domains, monolithic implementations are still effective, especially for applications where high efficiency, simplicity, and tight coupling are required. For instance, the Linux kernel, a monolithic system, continues to dominate operating system design, underpinning much of the world's critical computing infrastructure. As Thompson pointed out during the first debate, single-kernel implementations are simpler to implement and optimize for particular hardware platforms, which play a fundamental role in a vast number of performance-critical applications (DiBona & Ockman, 1999).

Finally, although microservices and microkernels have their strengths in terms of modularity and scalability, the fact that monolithic architectures such as Linux have been very successful also shows that both paradigms have their uses in contemporary computation. Instead of "confuting Torvalds," the emergence of microservices and microkernels highlights the breadth of architectural needs that exist across a variety of use cases. There are no substitutes for monolithic systems in applications in which performance, ease of use, and tight integration are of critical importance.

References

DiBona, C. & Ockman, S. (1999) Open Sources. 1st ed. Sebastapol: O’Reilly Media, Inc.

Fritzsch J., Bogner J., Zimmermann A. & Wagner S. (2019) From Monolith to Microservices: A Classification of Refactoring Approaches. In: Bruel JM., Mazzara M., Meyer B. (eds) Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. DEVOPS 2018. Lecture Notes in Computer Science, vol 11350. Springer.