TidBIT

Security cannot be an afterthought that is added to an application or system. It must be part of the design from the very beginning. Every method that can be executed, every user interface button or link, and every piece of data that can be returned should have a security requirement assigned to it. Frameworks such as Dropwizard, Spring, or even the built-in Java libraries have mechanisms that you can choose to use or not.

Part of the requirements gathering should be to identify the roles that users, including other programs that will call your code, are assigned to. You can decorate a REST endpoint with an attribute indicating the role required and call it finished. This is a "castle" mentality — the outer walls of the keep are strong and designed to keep unauthorized people out. But what if someone, or something, gets past that outer layer? If the entire code base behind that REST endpoint does not re-check permissions, then a coding mistake or flaw on that endpoint will leave the entire application bare and unprotected. Do not just code "happy path" logic assuming that whoever is calling a method can do so. Check first by passing the security token and role to inner methods so they have a chance to determine if the caller is authorized to its logic.

Required Resources

Reading: *Operating System Concepts (https://go.oreilly.com/SNHU/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_c14_r1.html)*, Chapter 14

(Sections 14.1, 14.2, 14.4, 14.6, 14.7), Chapter 15 (Sections 15.1–15.3, 15.5, 15.7)
Read the chapter sections in this eBook, which cover foundational concepts related to protection and security.