Sean Poston

CY201 Final

5/8/2020

Secure Software Engineering Principles

Software engineering is a broad field in the world of computers. It covers many different types of developers, from front-end developers, back-end developers, and security developers. However, no matter the type of developer one is, security is always of utmost importance. Using secure coding principles and practices is the hallmark of a good software engineer. There are many ways to do this: least privilege, fail-safe defaults, and open design are just a few principles to keep in mind on the job [1]. Security in software engineering is the backbone of security for everyone.

Least privilege is the “design principle [that] requires a minimalistic approach to granting user access rights to specific information and tools” [2]. What this means is that no one user should have any more privileges granted to them than they will absolutely need. While this can be frustrating for a user to continue asking someone with higher privileges than them to unlock something or do a simple task, the safety concerns that are avoided with this system is absolutely worth the hassle. By practicing least privilege rules, a software engineer will keep a whole system safer. Here is a sample diagram of least privilege at a business:

	Debbie in Accounting	Bill in Management	Josh in IT
Update	no	yes	yes
Insert	yes	yes	yes
Select	yes	yes	yes
Delete	no	no	yes
Grant	no	no	yes

Another security principle is keeping fail-safe defaults. Setting software to automatically assign users privileges as low or none is a basic, yet extremely effective method to security. Say a user is using MySQL on a network; when they are first given a MySQL account, the privileges are set to none. It takes a user with grant options to be able to give the new user privileges to be able to do anything. The alternative for this would be that a new user is given all privileges and another user would have to take away privileges until they have the desired level of access. This poses many problems, the most obvious being that the new user could easily wreak havoc (intentionally or not) if their privileges are not changed in time. It's for this reason that fail-safe defaults is a must have in any software.

Finally, the idea of open design posits that "the security of a system and its algorithms should not be depended on the secrecy of its design or implementation" [2]. Open design is showing your hand in a game of poker because you're sure you can still win. With open design, any user can find the inner workings of the code and algorithms. However, these users can't edit that code or any of the privileges granted to anyone [3]. The principle submits that knowing the software from the inside should not cause any security issues for the creators.

Security in software engineering is a broad field that cannot be covered in such a short paper. Here are only a few small, yet important, principles that every engineer, software, network, or otherwise, should be familiar with. These principles, and others, are the backbone for safety in the realm of computers. In a world where there's always someone who will want to do harm, it takes knowledge and practice from the software engineers to be able to counteract these users with malintent. While it is a daunting task, it is an achievable one if basic security principles are followed.

Works Cited

[1] “Developing Secure Software.” Davis, Noopur. *Secure Software Engineering.* https://www.csiac.org/wp-content/uploads/2016/02/stn8_2.pdf.

[2] “9 Software Security Design Principles.” Merritt, Todd. *DZone*. https://dzone.com/articles/9-software-security-design.

[3] “Design Principles for Security Mechanisms.” Bishop, Matt. *Conputer Security: Art and Science*. https://www.informit.com/articles/article.aspx?p=30487&seqNum=2