Kyle Anderson
Jvh640
CS 4683-001

8 Principles of Cyber Security

Saltzer and Schroeder's design principles are considered by many to be the foundation of building secure systems. Being written in 1975, these principles have done a fair job withstanding the test of time, most of which I and many others would consider to be still relevant today. In this synopsis, I will explain my thoughts on which of these is or is not relevant in today's tech industry.

The first design principle that Saltzer and Schroeder put forward was "Economy of Mechanism". Simply put this means: the simpler the system the easier it is to test, debug, and validate. I would argue that this is a principle that will never go out of style. The reason for this is because of the nature of complexity. Complexity and security seem to be natural inverses when looking at projects. Adding more features creates more complexity. More complexity creates problems for you to work through. I would classify this specific principle to be so fundamental, it could be considered the Occam's Razor of the technology industry.

History is riddled with failures of our next design principle where men and women's lives were paid in blood because of these failures. That is Fail-safe defaults. From the Challenger space shuttle's 'O' ring failure to the meltdown of not one, but two nuclear power plants in the last half century, the need for fail-safe defaults couldn't be more pronounced to human civilization you'd think. I'll let you be the judge on whether we have learned that lesson collectively as a civilization, but the premise of maintaining a default state that places our system in a position incapable of harm is just as sound today as it was almost 50 years ago.

Complete Mediation is our next design principle. This one is the idea that each time you access a system you're required to validate. As the last two have seemed fairly obvious to me I would argue that this is no different. Most of us have experienced the repercussions of this in some form or another at some point, be it a family member, friend, or coworker making an embarrassing post on social media when we step away from our device or us choosing to leave our systems unprotected. While these are examples of Complete Mediation failures made on the human side of the equation this failure can just as easily happen on the software side of the equation. Things like accessing a database using client side JS or simply not requiring validation at all can result in a data breach that can affect the identities of thousands or millions of people.

This next principle is probably my favorite and most preferred on this list: Open-Design. This principle is the concept of posting your software for public review and allowing an artificial selection process to take place. It is a fantastic way to expose bugs and vulnerabilities as the community using the project may potentially work their way to contributing to the product. It also provides more potential for peer review throughout the community. Open source projects tend to have longer life spans and be more reliable in my experience because of the dedicated communities behind them that continue to assist in the refinement process. This typically results in a more secure and reliable product over all. Additionally, this principle completely removes the temptation to take a "Security through obscurity" approach, which I would argue is the security equivalent of hiding from the Boogey man under a blanket.

Separation of Privilege is a principle that I would say is highly dependent on the potential repercussions of improper use. For systems that have the potential of life or limb it's a no brainer to maintain additional security measures. I could potentially see this being required for things like turning off the servers in the financial sector as minutes of downtime can translate to millions of dollars in lost revenue, or other truly life shattering repercussions. In most cases

though this type of system, I would consider it to be an unnecessary waste of time and resources as it will primarily slow production speed. This design principle I would consider an asset in cases of extreme risk.

The next principle is an ironclad principle that has withstood the test of time far beyond technology. Least Privilege in the technology sense originated in the mid 1960s, but the concept has been used by militaries around the world for so long that the exact timeframe isn't well documented. The principle maintains that a solid system never provides more privilege than is required to perform the tasks designated. This allows us to limit the potential risk of a disgruntled actor to cause damage to our system. Keeping this in mind it's also advised to perform regular security audits and maintain least privilege within an organization. Otherwise over a scope of time you could see significant privilege creep over time.

An area where the principle of Least Common Mechanism seems to have been forgotten or at least taken a major back seat is in cloud computing. Least Common Mechanism is the idea that you attempt to restrict the pathways used for communication or operation in the PC, utilizing separate CPUs, RAM, and Disks because of the potential of leakage outside our intended audience. In a cloud environment today this concept seems difficult in practice as most providers are slicing apart larger resources on shared systems into containers or hypervisors. This comes with the implicit risk that data could be leaked from one of these shared spaces. This being said, the reason cloud has grown in popularity is the potential to scale your products without having to purchase the physical and maintain your own data center or multiple datacenters. For smaller businesses this means less upfront cost at the expense of control. Least common mechanism is the principle here that I would say is the most underrated of the design principles suggested. The importance is clear but can easily be seen as a luxury rather than a necessity especially for smaller start ups.

Our final principle is Psychological Acceptability, and I would argue this is easily one of the most important and most difficult, as it specifically deals with the introduction of the human element, easily the most difficult component to control in any system. Simply put, people won't go out of their way to use security systems correctly if they don't agree with or understand the need for the said precautions. The truth is we're all guilty of this. This is why one of the most common passwords today is still 'password'. We understand that weak passwords grant easier access to unwanted parties, but we have a difficult time justifying the inconvenience of memorization or creation to protect our systems better. This expands far beyond the physical technology we deal with as well. Most companies frequently have seminars and trainings talking about the risk of social engineering, but when push comes to shove still today the biggest weakness at the vast majority of companies is social engineering, even after being educated on the topic. People implicitly want to be seen as helpful and liked which results in compliance even when the obvious is staring them in the face.

In conclusion, I think that Saltzer and Shcroeder's design principles are fairly ironclad. To me, most of these concepts seem obvious and can be inferred, I think that it is good to have them listed out as a standard. Interestingly enough though, while many would agree that most of these seem to be implicit, our standards still vary. The most obvious of them are still included such as least privilege, fail safe defaults, separation of privilege and least common mechanism in most of the standards with the addition of a few such as Defence in Depths. Truthfully, I don't

see these principles going anywhere in the future. They are applicable beyond tech and many have been implemented in some way shape or form in other industries, organizations, and processes since before the first computer was made.