The Saltzer and Schroeder List

Saltzer and Schroeder's 1976 paper listed eight design principles for computer security, and noted two additional principles that seemed relevant if more general.

1. Economy of mechanism – A simple design is easier to test and validate.

2. Fail-safe defaults – Figure 2 shows a physical example: outsiders can't enter a store via an emergency exit, and insiders may only use it in emergencies. In computing systems, the save default is generally “no access” so that the system must specifically grant access to resources. Most file access permissions work this way, though Windows also provides a “deny” right. Windows access control list (ACL) settings may be inherited, and the “deny” right gives the user an easy way to revoke a right granted through inheritance. However, this also illustrates why “default deny” is easier to understand and implement, since it's harder to interpret a mixture of “permit” and “deny” rights.

3. Complete mediation – Access rights are completely validated every time an access occurs. Systems should rely as little as possible on access decisions retrieved from a cache. Again, file permissions tend to reflect this model: the operating system checks the user requesting access against the file's ACL. The technique is less evident when applied to email, which must pass through separately applied packet filters, virus filters, and spam detectors.

4. Open design – Baran (1964) argued persuasively in an unclassified RAND report that secure systems, including cryptographic systems, should have unclassified designs. This reflects recommendations by Kerckhoffs (1883) as well as Shannon's maxim: “The enemy knows the system” (Shannon, 1948). Even the NSA, which resisted open crypto designs for decades, now uses the Advanced Encryption Standard to encrypt classified information.

5. Separation of privilege – A protection mechanism is more flexible if it requires two separate keys to unlock it, allowing for two-person control and similar techniques to prevent unilateral action by a subverted individual. The classic examples include dual keys for safety deposit boxes and the two-person control applied to nuclear weapons and Top Secret crypto materials. Figure 3 (courtesy of the Titan Missile Museum) shows how two separate padlocks were used to secure the launch codes for a Titan nuclear missile.Separation of privilege – A protection mechanism is more flexible if it requires two separate keys to unlock it, allowing for two-person control and similar techniques to prevent unilateral action by a subverted individual. The classic examples include dual keys for safety deposit boxes and the two-person control applied to nuclear weapons and Top Secret crypto materials.

6. Least privilege – Every program and user should operate while invoking as few privileges as possible. This is the rationale behind Unix “sudo” and Windows User Account Control, both of which allow a user to apply administrative rights temporarily to perform a privileged task.

7. Least common mechanism – Users should not share system mechanisms except when absolutely necessary, because shared mechanisms may provide unintended communication paths or means of interference.

8. Psychological acceptability – This principle essentially requires the policy interface to reflect the user's mental model of protection, and notes that users won't specify protections correctly if the

specification style doesn't make sense to them.

There were also two principles that Saltzer and Schroeder noted as being familiar in physical security but applying “imperfectly” to computer systems:

1. Work factor – Stronger security measures pose more work for the attacker. The authors acknowledged that such a measure could estimate trial-and-error attacks on randomly chosen passwords. However, they questioned its relevance since there often existed “indirect strategies” to penetrate a computer by exploiting flaws. “Tiger teams” in the early 1970s had systematically found flaws in software systems that allowed successful penetration, and there was not yet enough experience to apply work factor estimates effectively.

2. Compromise recording – The system should keep records of attacks even if the attacks aren't necessarily blocked. The authors were skeptical about this, since the system ought to be able to prevent penetrations in the first place. If the system couldn't prevent a penetration or other attack, then it was possible that the compromise recording itself may be modified or destroyed.

Today, of course, most analysts and developers embrace these final two design principles. The argument underlying complex password selection reflect a work factor calculation, as do the recommendations on choosing cryptographic keys. Compromise recording has become an essential feature of every secure system in the form of event logging and auditing.