4th Design Decision Should the tasks of defining and enforcing security be given to a central entity or should they be left to individual components in a system?

This question arises naturally in distributed systems security and you will see examples of both alternatives. However, this question is also meaningful in the context of mainframe systems as demonstrated by the mandatory and discretionary security policies of the Bell–LaPadula model covered in section 8.2.

2.5 THE LAYER BELOW

So far, we have briefly touched on assurance but have predominantly explored options for expressing the most appropriate security policies. It is now time to think about attackers trying to bypass our protection mechanisms. Every protection mechanism defines a *security perimeter (boundary)*. The parts of the system that can malfunction without compromising the protection mechanism lie outside this perimeter. The parts of the system that can be used to disable the protection mechanism lie within this perimeter. This observation leads to an immediate and important extension of the second design decision proposed in section 2.4.2.

5th Design Decision How can you prevent an attacker getting access to a layer below the protection mechanism?

An attacker with access to the 'layer below' is in a position to subvert protection mechanisms further up. For example, if you gain systems privileges in the operating system, you are usually able to change programs or files containing the control data for security mechanisms in the services and applications layers. If you have direct access to the physical memory devices so that you can manipulate the raw data, the logical access controls of the operating system have been bypassed. Below, we give six further examples to illustrate this point. The fact that security mechanisms have a soft underbelly and are vulnerable to attacks from lower layers should be a reason for concern, but not for despair. When you reach the stage where you cannot apply computer security mechanisms or do not want to do so, you can still put in place physical or organizational security measures (Figure 2.6).

Recovery Tools

If the logical organization of the memory is destroyed due to some physical memory fault, it is no longer possible to access files even if their physical representation is still intact. Recovery tools, like Norton Utilities, can help to restore the data by reading the (physical) memory directly and then restoring the file structure. Such a tool can, of course, be used to circumvent logical access control as it does not care for the logical memory structure.

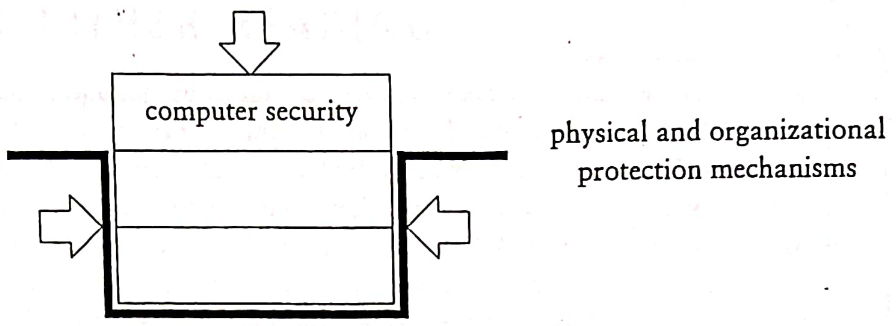

Figure 2.6: Physical and Organizational Security Measures Controlling Access to the Layers Below

Unix Devices

Unix treats I/O devices and physical memory devices like files. The same access control mechanisms can therefore be applied to these devices as to files. If access permissions are defined badly, e.g. if read access is given to a disk which contains read-protected files, then an attacker can read the disk contents and reconstruct the files. You can find more information on Unix security in Chapter 6.

Object Reuse (Release of Memory)

A single-processor multiprogramming system may execute several processes at the same time but only one process can 'own' the processor at any point in time. Whenever the operating system deactivates the running process to activate the next, a *context switch* is performed. All data necessary for the later continuation of the execution is saved and memory is allocated for the new process. *Storage residues* are data left behind in the memory area allocated to the new process. If the new process could read such storage residues, the logical separation between processes that the operating system should provide has been breached. To avoid this problem, all memory locations that are released could be overwritten with a fixed pattern or the new process could be granted read access only to locations it has already written to.

Buffer Overruns

In a buffer overrun attack, a value is assigned to a variable that is too large for the memory buffer allocated to that variable, so that memory allocated to other variables is overwritten. This method for modifying variables that should be logically inaccessible is further explained in section 14.4.1.

Backup

A conscientious system manager will perform regular backups. Whoever can lay their hands on the backup tape has access to all the data on the tape and logical access control is of no help. Thus, backup tapes have to be locked away safely to protect the data.

Core Dumps

When a system crashes, it creates a *core dump* of its internal state so that the reasons for the crash can be more easily identified. If the internal state contains sensitive information, like cryptographic keys, and if core dumps are stored in files that can be read by anyone, an attacker could intentionally crash a multiuser system and look in the core dump for data belonging to other users.

2.6 FURTHER READING

To get a second opinion on computer security, there are a number of books you could consult. A very readable introduction to the subject is provided by Russel and Gangemi Sr. (1991). Amoroso (1994) covers the theoretical elements of computer security. A discussion of the technicalities of designing secure operating systems from the 1980s can be found in Gasser (1988) (out of print but available on the Web). Another comprehensive treatment of information security, with many valuable pointers for further reading, is presented by Pfleeger and Lawrence Pfleeger (2003).

Books on the security features of specific operating systems tend to be rather expensive, concentrate on issues relevant to someone managing such a system, like the menus to call up and the options to choose from, but do not provide too much further insight into the way security is implemented. Notable exceptions to this rule are Park's book on AS/400 (1995), which goes into technical details of the operating system and shows how code at lower layers can compromise the security provided by the operating system, and Brown's book on Windows security (2000). Good books on Unix security are by Curry (1992), Ferbrache and Shearer (1992) and Garfinkel, Spafford and Schwartz (2003). For more specific topics, you will find further pointers to sources at the end of the relevant chapters.

To appreciate the current state of an area of research, you should also be aware of its history. Two influential reports that triggered much research in computer security are by Ware (1979) and Anderson (1972). If you cannot get hold of these reports, MacKenzie and Pottinger (1997) provide a summary of the early history of computer security.