The processes in an operating system must be protected from one another’s activities. To provide such protection, we can use various mechanisms to ensure that only processes that have gained proper authorization from the operating system can operate on the files, memory segments, CPU, and other resources of a system.

Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system. This mechanism must provide a means for specifying the controls to be imposed, together with a means of enforcement. We distinguish between protection and security, which is a measure of confidence that the integrity of a system and its data will be preserved. In this chapter, we focus on protection. Security assurance is a much broader topic, and we address it in Chapter 14.

Bibliographical Notes

The access-matrix model of protection between domains and objects was developed by [Lampson (1969)] and [Lampson (1971)]. [Popek (1974)] and [Saltzer and Schroeder (1975)] provided excellent surveys on the subject of protection. [Harrison et al. (1976)] used a formal version of this model to enable them to prove properties of a protection system mathematically.

The concept of a capability evolved from Iliffe’s and Jodeit’s codewords, which were implemented in the Rice University computer ([Iliffe and Jodeit (1962)]). The term capability was introduced by [Dennis and Horn (1966)].

The Hydra system was described by [Wulf et al. (1981)]. The CAP system was described by [Needham and Walker (1977)]. [Organick (1972)] discussed the MULTICS ring-protection system.

Revocation was discussed by [Redell and Fabry (1974)], [Cohen and Jefferson (1975)], and [Ekanadham and Bernstein (1979)]. The principle of separation of policy and mechanism was advocated by the designer of Hydra ([Levin et al. (1975)]). The confinement problem was first discussed by [Lampson (1973)] and was further examined by [Lipner (1975)].

The use of higher-level languages for specifying access control was suggested first by [Morris (1973)], who proposed the use of the seal and
unseal operations discussed in Section 13.9. [Kieburtz and Silberschatz (1978)], [Kieburtz and Silberschatz (1983)], and [McGraw and Andrews (1979)] proposed various language constructs for dealing with general dynamic-resource-management schemes. [Jones and Liskov (1978)] considered how a static access-control scheme can be incorporated in a programming language that supports abstract data types. The use of minimal operating-system support to enforce protection was advocated by the Exokernel Project ([Ganger et al. (2002)], [Kaashoek et al. (1997)]). Extensibility of system code through language-based protection mechanisms was discussed in [Bershad et al. (1995)]. Other techniques for enforcing protection include sandboxing ([Goldberg et al. (1996)]) and software fault isolation ([Wahbe et al. (1993)]). The issues of lowering the overhead associated with protection costs and enabling user-level access to networking devices were discussed in [McCanne and Jacobson (1993)] and [Basu et al. (1995)].

More detailed analyses of stack inspection, including comparisons with other approaches to Java security, can be found in [Wallach et al. (1997)] and [Gong et al. (1997)].

Bibliography


Chapter 13  Protection


