Protection

Practice Exercises

17.1 What are the main differences between capability lists and access lists?

Answer:
An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object. A capability list is a list of objects and the operations allowed on those objects for each domain.

17.2 A Burroughs B7000/B6000 MCP file can be tagged as sensitive data. When such a file is deleted, its storage area is overwritten by some random bits. For what purpose would such a scheme be useful?

Answer:
This would be useful as an extra security measure so that the old content of memory cannot be accessed, either intentionally or by accident, by another program. This is especially useful for any highly classified information.

17.3 In a ring-protection system, level 0 has the greatest access to objects, and level n (where $n > 0$) has fewer access rights. The access rights of a program at a particular level in the ring structure are considered a set of capabilities. What is the relationship between the capabilities of a domain at level j and a domain at level i to an object (for $j > i$)?

Answer:
D_j is a subset of D_i.

17.4 The RC 4000 system, among others, has defined a tree of processes (called a process tree) such that all the descendants of a process can be given resources (objects) and access rights by their ancestors only. Thus, a descendant can never have the ability to do anything that its ancestors cannot do. The root of the tree is the operating system, which has the ability to do anything. Assume that the set of access rights is represented by an access matrix, A. $A(x,y)$ defines the access rights of process x to object y. If x is a descendant of z, what is the relationship between $A(x,y)$ and $A(z,y)$ for an arbitrary object y?

Answer:
$A(x,y)$ is a subset of $A(z,y)$.

17.5 What protection problems may arise if a shared stack is used for parameter passing?

Answer:
The contents of the stack could be compromised by any other processes sharing the stack.

17.6 Consider a computing environment where a unique number is associated with each process and each object in the system. Suppose that we allow a process with number n to access an object with number m only if $n > m$. What type of protection structure do we have?

Answer:
A hierarchical structure.

17.7 Consider a computing environment where a process is given the privilege of accessing an object only n times. Suppose a scheme for implementing this policy.

Answer:
Add an integer counter with the capability.

17.8 If all the access rights to an object are deleted, the object can no longer be accessed. At this point, the object should also be deleted, and the space it occupies should be returned to the system. Suggest an efficient implementation of this scheme.

Answer:
Reference counts.

17.9 Why is it difficult to protect a system in which users are allowed to do their own I/O?

Answer:
In earlier chapters, we identified a distinction between kernel and user mode whereby kernel mode is used for carrying out privileged operations such as I/O. One reason why I/O must be performed in kernel mode is that I/O requires accessing the hardware, and proper access to the hardware is necessary for system integrity. If we allow users to perform their own I/O, we cannot guarantee system integrity.

17.10 Capability lists are usually kept within the address space of the user. How does the system ensure that the user cannot modify the contents of the list?

Answer:
A capability list is considered a "protected object" and is accessed only indirectly by the user. The operating system ensures that the user cannot access the capability list directly.

Exercises

17.11 The access-control matrix can be used to determine whether a process can switch from, say, domain A to domain B and enjoy the access privileges of domain B. Is this approach equivalent to including the access privileges of domain B in those of domain A?

Answer:
Yes, this approach is equivalent to including the access privileges of domain B in those of domain A as long as the switch privileges associated with domain B are also copied over to domain A.

17.12 What hardware features does a computer system need for efficient capability manipulation? Can these features be used for memory protection?

Answer:
A hardware feature is needed allowing a capability object to be identified as either a capability or accessible object. Typically, several bits are necessary to distinguish between different types of capability objects. For example, 4 bits could be used to uniquely identify 2^4 or 16 different types of capability objects.

These could not be used for routine memory protection, as they offer little for protection apart from a binary value indicating whether they are a capability object or not. Memory protection requires full support from virtual memory features discussed in Chapter 10.

17.13 Discuss the strengths and weaknesses of implementing an access matrix using capabilities that are associated with domains.

Answer:
Capabilities associated with domains provide substantial flexibility and faster access to objects. When a domain presents a capability, the system just needs to check the authenticity of the capability, and that can be performed efficiently. Capabilities can also be passed around from one domain to another domain with great ease, further contributing to flexibility. However, the flexibility comes at the cost of a lack of control: revoking capabilities and restricting the flow of capabilities are difficult tasks.

17.14 What is the need-to-know principle? Why is it important for a protection system to adhere to this principle?

Answer:
A process may access at any time those resources that it has been authorized to access *and* that are currently required to complete its task. It is important in that it limits the amount of damage a faulty process can cause in a system.

17.15 Describe how the Java protection model would be compromised if a Java program were allowed to directly alter the annotations of its stack frame.

Answer:

When a thread issues an access request in a `doPrivileged()` block, the stack frame of the calling thread is *annotated* according to the calling thread's protection domain. A thread with an annotated stack frame can make subsequent method calls that require certain privileges. Thus, the annotation serves to mark a calling thread as being privileged. If a Java program were allowed to directly alter the annotations of a stack frame, it could potentially perform an operation for which it did not have the necessary permissions, thus violating the security model of Java.

17.16 How does the principle of least privilege aid in the creation of protection systems?

Answer:

The principle of least privilege allows users to be given just enough privileges to perform their tasks. When a system is implemented within the framework of this principle, a failure or compromise of a component does the minimum damage to the system, since the failed or compromised component has the smallest set of privileges required to support its normal mode of operation.