12

Security

Learning Objectives

In this chapter, we study the principles of engineering secure distributed objects. We will understand the potential security threats to which distributed objects are exposed. We will then learn about the principles of encryption and recognize that they provide the basic primitives upon which higher levels of security are built. We will comprehend these higher levels, which are concerned with the authenticity of objects, restriction of requests to authorized clients, auditing of security-relevant events, and provision of non-repudiable evidence. We investigate how higher-levels of security are supported by the CORBA Security service.

Chapter Outline

In the previous chapters we have started to address how reliable distributed objects can be built. We have investigated how to store the state of objects onto persistent storage so that it is retained after deliberate or accidental termination of server objects. We have also seen how transactions are used to limit the impact of failures. Reliability, however has a different facet that we have neglected so far. Using object-oriented middleware, we can build server objects and make them available to clients anywhere on a local area network, or even the Internet. This means that every client that can obtain an object reference to a server object can request operation execution from the server. Very often it is necessary to restrict access to a more limited community of client objects, to allow client objects only to execute particular operations, to secure the parameters of object requests against unauthorized access while in transit on a public network, to provide non-repudiable evidence that a client has indeed requested execution of a particular operation, and to record any incidents that may compromise the overall system security. We continue our banking example from the previous chapter in Example 12.1 to illustrate this point.

Example 12.1
Why We Need Security in Distributed Object Systems

Consider the availability of server objects on the Internet that is necessitated by a direct banking application. A bank will have to keep tight control over the client objects that it permits to execute credit or debit operations on account objects. The bank will also have to be convinced that the principal who executes the client object is actually who he or she claims to be. Moreover, the bank will have to provide non-repudiable evidence that the principal requested a particular operation. In the Internet direct banking application, the bank will have to generate evidence that the principal executing the client has, for example, requested a funds transfer. Finally, the auditors of the bank will demand from the application that it provides details about any attack or other security-relevant event that occurred within the system. All these requirements demand the provision of high-level primitives for security.

In order to understand the importance of security and the measures against which distributed systems have to be secured, we discuss the different security threats and the methods that are used for attacking security. The higher levels of security we ultimately want to achieve are implemented using encryption techniques. We then review public and secure key-based encryption methods and the key distribution techniques that are associated with them. We review high-level security services, such as authentication, access control, non-repudiation and auditing, that are built on top of the encryption primitives. Finally, we review the security services provided by object-oriented middleware to provide examples of how these principles are applied.

12.1 Security Attacks

12.1.1 Insecurity of Distributed Systems

The components of distributed systems, whether object-oriented or not, have to communicate via a network. If hosts are directly connected to public networks, such as the Internet, any network traffic to or from the host is publicly accessible. Individuals wishing to launch

security attacks can obtain connections to a public network, because there are no restrictions on connecting machines to a public network. This means that an attacker can use a public network to send messages to distributed system components in an attempt to compromise the system security.

An obvious approach to avoiding security problems is to physically separate the network from the public network. These separated networks are called private networks and are still often employed by banks and telecommunications operators to secure their mission-critical systems. As a consequence of such a separation, however, we cannot have distributed system communication across the two networks. With the advent of electronic commerce, distributed system components often cannot have such a physical separation and even critical systems have to be connected to public networks. Even when networks are private, or the connection between hosts on a private network and hosts on a public network is tightly controlled, security can be compromised. Individuals who have authorized access to machines connected to the private network may use these machines to launch security attacks.

Distributed systems are inherently insecure.

This leads us to the observation that distributed systems are inherently insecure. If an attacker succeeds in breaking the security of a system, the legitimate users will lose confidence in the system. They may sue the engineers or operators of the system for the damages that were caused as a result of the security attack. For the engineering of distributed objects this means that one should not trust any other distributed object, because it may be written by an attacker, who wants to compromise security. We thus have to engineer both client and server objects in such a secure way that confidence and trust can be re-established. In order to do this, we first have to develop a better understanding of the security threats to which distributed objects are exposed.

12.1.2 Threats

There are different forms of security threat. [Colouris et al., 1994] classify them into four different categories and we follow this classification. The first class is concerned with leakage of information to non-authorized parties. The second class includes threats of non-authorized parties modifying information that is stored or manipulated in a distributed system. The third class is concerned with the unauthorized use of resources, such as network bandwidth, disk space or CPU time. The last, and probably most serious, threat involves the destruction of information or the distributed system itself by unauthorized parties. We now discuss each of these threats and provide examples that show why they can be very serious.

Leakage

Leakage is the unauthorized disclosure of information.

A *leakage* is the disclosure of information to unauthorized parties. The damage that can be caused through leakage greatly varies depending on the nature of the information and the relationship between the attacker and the owner of the information. Leakage may lead to loss of business: if videos are leaked from a video-on-demand server then attackers can watch the videos without paying. Leakage may also lead to a loss of privacy, for example, if the financial standing of an account owner is disclosed to third parties. Leakage may also have effects that are more damaging than the loss of privacy.

Example 12.2
Leakage of Account Information

The balance of a personal or company account is considered to be private to that person. Yet, sometimes it would be advantageous for competitor companies or companies planning to buy a company to have detailed insights into account balances. If a company manages to hack into the bank database and obtain this information, this is an example of leakage of information.

Tampering

Tampering is unauthorized modification of information.

Tampering denotes the unauthorized modification of information that is stored or manipulated in a distributed system. Tampering with information implies leakage but goes beyond it, in that the information is not only accessed but also modified. Tampering is potentially a more dangerous threat than leakage, because the attacker modifies an organization's perception of the world as it is captured in the organization's data.

Example 12.3
Tampering with Bank Accounts

Attackers that can tamper with the information that is held about accounts in a distributed system of a bank can effectively modify account balances. They can, for example, increase the balances of their own accounts to increase the funds that they can command.

Resource Stealing

Resource stealing denotes the unauthorized use of computing resources.

The class of threats that are subsumed under *resource stealing* use the resources provided by the distributed system in an unauthorized way. Such unauthorized usage often coincides with usage for which the attackers are not paying, which is why we refer to the use as stealing. Stolen resources can be CPU time, main memory, network bandwidth, air-time of mobile phones, secondary storage memory or even human resources who operate a computer system.

Example 12.4
Resource Stealing

Telephone network management applications are often distributed systems. By violating the security of these distributed systems, attackers may be able to use the resources of a telephone network provider without paying. They may, for example, attempt to modify the distributed billing system of the provider in such a way that they can use their telephone without being charged.

Vandalism

Vandalism is the destruction of information.

Vandalism is potentially the most serious threat and in the context of distributed system security it denotes the destruction of information stored by distributed objects or the distributed object system itself.

Example 12.5
Vandalism

There have been examples where attackers obtained administrator rights and then formatted hard disks and thus destroyed data and code alike.

Combined Threats

Attackers very often combine the different threats discussed above. They steal resources such as telephone network access in order to connect themselves to a public network. They then attempt to obtain sensitive information, such as passwords or login details. Once they obtain those, they login to a machine and launch security attacks using another user's identity and may tamper with data in a way that is beneficial to them.

12.1.3 Methods of Attack

Now that we have understood the different forms of security threats that distributed object systems are exposed to, we can investigate the methods that attackers use to exercise these threats. Understanding these methods is important in order to engineer distributed objects in such a way that these methods cannot be applied.

The methods can be classified into four categories. Methods in the first category involve concealing the attacker's identity so that they can pretend to be an authorized user of a distributed object. The second category involves listening to network traffic in order to obtain the contents of object requests. The third category includes methods that modify object requests. And the last category includes methods that are based on repeating requests that have previously been exchanged between distributed objects. We now discuss each of these methods in more detail.

Masquerading

Masquerading is a method of attack where the attacker obtains the identity of a different, usually an authorized, user. Masquerading is also sometimes referred to as *spoofing*. The attacker can then use the access rights and privileges of that user to perform malicious actions. Identity in centralized systems is usually validated at session startup time. In distributed object systems, however, the identity has to be passed and validated with each object request.

Masquerading attacks obtain the identity of legitimate users.

Example 12.6
Masquerading Attack on a Bank Account

The operations to credit and debit accounts in our banking example should obviously only be executed by authorized personnel of the bank in such a way that there is never a credit without a debit operation. A secure implementation of the banking example would have to distinguish different groups of users and grant the right to execute the `credit` and `debit` operations only to a particular group. An attacker wishing to credit monies on his account would then have to use masquerading to obtain the identity of one of the authorized users.

In order to prevent attackers from using masquerading, we will have to make it difficult for them to obtain the identity of authorized users. *Authentication* is used for that purpose. Authentication supports proving that a user is who the user claims to be. The implementation of authentication is based on encryption techniques, which we will study in the next section.