1.1. The Basic Components

Computer security rests on confidentiality, integrity, and availability. The interpretations of these three aspects vary, as do the contexts in which they arise. The interpretation of an aspect in a given environment is dictated by the needs of the individuals, customs, and laws of the particular organization.

1.1.1. Confidentiality

Confidentiality is the concealment of information or resources. The need for keeping information secret arises from the use of computers in sensitive fields such as government and industry. For example, military and civilian institutions in the government often restrict access to information to those who need that information. The first formal work in computer security was motivated by the military's attempt to implement controls to enforce a “need to know” principle. This principle also applies to industrial firms, which keep their proprietary designs secure lest their competitors try to steal the designs. As a further example, all types of institutions keep personnel records secret.

Access control mechanisms support confidentiality. One access control mechanism for preserving confidentiality is cryptography, which scrambles data to make it incomprehensible. A **cryptographic key** controls access to the unscrambled data, but then the cryptographic key itself becomes another datum to be protected.

> EXAMPLE: Enciphering an income tax return will prevent anyone from reading it. If the owner needs to see the return, it must be deciphered. Only the possessor of the cryptographic key can enter it into a deciphering program. However, if someone else can read the key when it is entered into the program, the confidentiality of the tax return has been compromised.

Other system-dependent mechanisms can prevent processes from illicitly accessing information. Unlike enciphered data, however, data protected only by these controls can be read when the controls fail or are bypassed. Then their advantage is offset by a corresponding disadvantage. They can protect the secrecy of data more completely than cryptography, but if they fail or are evaded, the data becomes visible.

Confidentiality also applies to the existence of data, which is sometimes more revealing than the data itself. The precise number of people who distrust a politician may be less important than knowing that such a poll was taken by the politician's staff. How a particular government agency harassed citizens in its country may be less important than knowing that such harassment occurred. Access control mechanisms sometimes conceal the mere existence of data, lest the existence itself reveal information that should be protected.

Resource hiding is another important aspect of confidentiality. Sites often wish to conceal their configuration as well as what systems they are using; organizations may not wish others to know about specific equipment (because it could be used without authorization or in inappropriate ways), and a company renting time from a service provider may not want others to know what resources it is using. Access control mechanisms provide these capabilities as well.

All the mechanisms that enforce confidentiality require supporting services from the system. The assumption is that the security services can rely on the kernel, and other agents, to supply correct data. Thus, assumptions and trust underlie confidentiality mechanisms.

1.1.2. Integrity

Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Integrity includes data integrity (the content of the information) and origin integrity (the source of the data, often called **authentication**). The source of the information may bear on its accuracy and credibility and on the trust that people place in the information. This dichotomy illustrates the principle that the aspect of integrity known as credibility is central to the proper functioning of a system. We will return to this issue when discussing malicious logic.

EXAMPLE: A newspaper may print information obtained from a leak at the White House but attribute it to the wrong source. The information is printed as received (preserving data integrity), but its source is incorrect (corrupting origin integrity).

Integrity mechanisms fall into two classes: **prevention** mechanisms and **detection** mechanisms.

Prevention mechanisms seek to maintain the integrity of the data by blocking any unauthorized attempts to change the data or any attempts to change the data in unauthorized ways. The distinction between these two types of attempts is important. The former occurs when a user tries to change data which she has no authority to change. The latter occurs when a user authorized to make certain changes in the data tries to change the data in other ways. For example, suppose an accounting system is on a computer. Someone breaks into the system and tries to modify the accounting data. Then an unauthorized user has tried to violate the integrity of the accounting database. But if an accountant hired by the firm to maintain its books tries to embezzle money by sending it overseas and hiding the transactions, a user (the accountant) has tried to change data (the accounting data) in unauthorized ways (by moving it to a Swiss bank account). Adequate authentication and access controls will generally stop the break-in from the outside, but preventing the second type of attempt requires very different controls.

Detection mechanisms do not try to prevent violations of integrity; they simply report that the data's integrity is no longer trustworthy. Detection mechanisms may analyze system events (user or system actions) to detect problems or (more commonly) may analyze the data itself to see if required or expected constraints still hold. The mechanisms may report the actual cause of the integrity violation (a specific part of a file was altered), or they may simply report that the file is now corrupt.

Working with integrity is very different from working with confidentiality. With confidentiality, the data is either compromised or it is not, but integrity includes both the correctness and the trustworthiness of the data. The origin of the data (how and from whom it was obtained), how well the data was protected before it arrived at the current machine, and how well the data is protected on the current machine all affect the integrity of the data. Thus, evaluating integrity is often very difficult, because it relies on assumptions about the source of the data and about trust in that source—two underpinnings of security that are often overlooked.

1.1.3. Availability

Availability refers to the ability to use the information or resource desired. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all. The aspect of availability that is relevant to security is that someone may deliberately arrange to deny access to data or to a service by making it unavailable. System designs usually assume a statistical model to analyze expected patterns of use, and mechanisms ensure availability when that statistical model holds. Someone may be able to manipulate use (or parameters that control use, such as network traffic) so that the assumptions of the statistical model are no longer valid. This means that the mechanisms for keeping the resource or data available are working in an environment for which they were not designed. As a result, they will often fail.

EXAMPLE: Suppose Anne has compromised a bank's secondary system server, which supplies bank account balances. When anyone else asks that server for information, Anne can supply any information she desires. Merchants validate checks by contacting the bank's primary balance server. If a merchant gets no response, the secondary server will be asked to supply the data. Anne's colleague prevents merchants from contacting the primary balance server, so all merchant queries go to the secondary server. Anne will never have a check turned down, regardless of her actual account balance. Notice that if the bank had only one server (the primary one), this scheme would not work. The merchant would be unable to validate the check.

Attempts to block availability, called **denial of service attacks**, can be the most difficult to detect, because the analyst must determine if the unusual access patterns are attributable to deliberate manipulation of resources or of environment. Complicating this determination is the nature of statistical models. Even if the model accurately describes the environment, atypical events simply contribute to the nature of the statistics. A deliberate attempt to make a resource unavailable may simply look like, or be, an atypical event. In some environments, it may not even appear atypical.