4.1. Security Policies

Consider a computer system to be a finite-state automaton with a set of transition functions that change state. Then:

> **Definition 4–1.** A **security policy** is a statement that partitions the states of the system into a set of **authorized**, or **secure**, states and a set of **unauthorized**, or **nonsecure**, states.

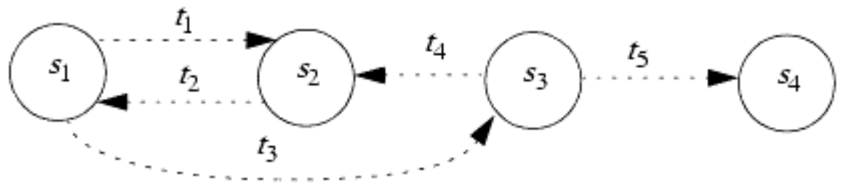
A security policy sets the context in which we can define a secure system. What is secure under one policy may not be secure under a different policy. More precisely:

> **Definition 4–2.** A **secure system** is a system that starts in an authorized state and cannot enter an unauthorized state.

Consider the finite-state machine in Figure 4-1. It consists of four states and five transitions. The security policy partitions the states into a set of authorized states **A** = { $\mathbf{s}_1$, $\mathbf{s}_2$ } and a set of unauthorized states **UA** = { $\mathbf{s}_3$, $\mathbf{s}_4$ }. This system is not secure, because regardless of which authorized state it starts in, it can enter an unauthorized state. However, if the edge from $\mathbf{s}_1$ to $\mathbf{s}_3$ were not present, the system would be secure, because it could not enter an unauthorized state from an authorized state.

> **Definition 4–3.** A **breach of security** occurs when a system enters an unauthorized state.

Figure 4-1. A simple finite-state machine. In this example, the authorized states are s_1 and s_2.



We informally discussed the three basic properties relevant to security in Section 1.1. We now define them precisely.

> **Definition 4–4.** Let **X** be a set of entities and let **I** be some information. Then **I** has the property of **confidentiality** with respect to **X** if no member of **X** can obtain information about **I**.

Confidentiality implies that information must not be disclosed to some set of entities. It may be disclosed to others. The membership of set **X** is often implicit—for example, when we speak of a document that is confidential. Some entity has access to the document. All entities not authorized to have such access make up the set **X**.

> **Definition 4–5.** Let **X** be a set of entities and let **I** be some information or a resource. Then **I** has the property of **integrity** with respect to **X** if all members of **X** trust **I**.

This definition is deceptively simple. In addition to trusting the information itself, the members of **X** also trust that the conveyance and storage of **I** do not change the information or its trustworthiness (this aspect is sometimes called **data integrity**). If **I** is information about the origin of something, or about an identity, the members of **X** trust that the information is correct and unchanged (this aspect is sometimes called **origin integrity** or, more commonly, **authentication**). Also, **I** may be a resource rather than information. In that case, integrity means that the resource functions correctly (meeting its specifications). This aspect is called **assurance** and will be discussed in Part 6, "Assurance." As with confidentiality, the membership of **X** is often implicit.

> **Definition 4–6.** Let **X** be a set of entities and let **I** be a resource. Then **I** has the property of **availability** with respect to **X** if all members of **X** can access **I**.

The exact definition of "access" in Definition 4–6 varies depending on the needs of the members of **X**, the nature of the resource, and the use to which the resource is put. If a book-selling server takes up to 1 hour to service a request to purchase a book, that may meet the client's requirements for "availability." If a server of medical information takes up to 1 hour to service a request for information regarding an allergy to an anesthetic, that will not meet an emergency room's requirements for "availability."

A security policy considers all relevant aspects of confidentiality, integrity, and availability. With respect to confidentiality, it identifies those states in which information leaks to those not authorized to receive it. This includes not only the leakage of rights but also the illicit transmission of information without leakage of rights, called **information flow**. Also, the policy must handle dynamic changes of authorization, so it includes a temporal element. For example, a contractor working for a company may be authorized to access proprietary information during the lifetime of a nondisclosure agreement, but when that nondisclosure agreement expires, the contractor can no longer access that information. This aspect of the security policy is often called a **confidentiality policy**.

With respect to integrity, a security policy identifies authorized ways in which information may be altered and entities authorized to alter it. Authorization may derive from a variety of relationships, and external influences may constrain it; for example, in many transactions, a principle called **separation of duties** forbids an entity from completing the transaction on its own. Those parts of the security policy that describe the conditions and manner in which data can be altered are called the **integrity policy**.

With respect to availability, a security policy describes what services must be provided. It may present parameters within which the services will be accessible—for example, that a browser may download Web pages but not Java applets. It may require a level of service—for example, that a server will provide authentication data within 1 minute of the request being made. This relates directly to issues of quality of service.

The statement of a security policy may formally state the desired properties of the system. If the system is to be provably secure, the formal statement will allow the designers and implementers to prove that those desired properties hold. If a formal proof is unnecessary or infeasible, analysts can test that the desired properties hold for some set of inputs. Later chapters will discuss both these topics in detail.

In practice, a less formal type of security policy defines the set of authorized states. Typically, the security policy assumes that the reader understands the context in which the policy is issued—in particular, the laws, organizational policies, and other environmental factors. The security policy then describes conduct, actions, and authorizations defining "authorized users" and "authorized use."

EXAMPLE: A university disallows cheating, which is defined to include copying another student's homework assignment (with or without permission). A computer science class requires the students to do their homework on the department's computer. One student notices that a second student has not read protected the file containing her homework and copies it. Has either student (or have both students) breached security?

The second student has not, despite her failure to protect her homework. The security policy requires no action to prevent files from being read. Although she may have been too trusting, the policy does not ban this; hence, the second student has not breached security.

The first student has breached security. The security policy disallows the copying of homework, and the student has done exactly that. Whether the security policy specifically states that "files containing homework shall not be copied" or simply says that "users are bound by the rules of the university" is irrelevant; in the latter case, one of those rules bans cheating. If the security policy is silent on such matters, the most reasonable interpretation is that the policy disallows actions that the university disallows, because the computer science department is part of the university.

The retort that the first user could copy the files, and therefore the action is allowed, confuses **mechanism** with **policy**. The distinction is sharp:

Definition 4–7. A **security mechanism** is an entity or procedure that enforces some part of the security policy.

EXAMPLE: In the preceding example, the policy is the statement that no student may copy another student's homework. One mechanism is the file access controls; if the second student had set permissions to prevent the first student from reading the file containing her homework, the first student could not have copied that file.

EXAMPLE: Another site's security policy states that information relating to a particular product is proprietary and is not to leave the control of the company. The company stores its backup tapes in a vault in the town's bank (this is common practice in case the computer installation is completely destroyed). The company must ensure that only authorized employees have access to the backup tapes even when the tapes are stored off-site; hence, the bank's controls on access to the vault, and the procedures used to transport the tapes to and from the bank, are considered security mechanisms. Note that these mechanisms are not technical controls built into the computer. Procedural, or operational, controls also can be security mechanisms.

Security policies are often implicit rather than explicit. This causes confusion, especially when the policy is defined in terms of the mechanisms. This definition may be ambiguous—for example, if some mechanisms prevent a specific action and others allow it. Such policies lead to confusion, and sites should avoid them.

EXAMPLE: The UNIX operating system, initially developed for a small research group, had mechanisms sufficient to prevent users from accidentally damaging one another's files; for example, the user **ken** could not delete the user **dmr**'s files (unless **dmr** had set the files and the containing directories appropriately). The implied security policy for this friendly environment was “do not delete or corrupt another's files, and any file not protected may be read.”

When the UNIX operating system moved into academic institutions and commercial and government environments, the previous security policy became inadequate; for example, some files had to be protected from individual users (rather than from groups of users). Not surprisingly, the security mechanisms were inadequate for those environments.

The difference between a policy and an abstract description of that policy is crucial to the analysis that follows.

Definition 4–8. A **security model** is a model that represents a particular policy or set of policies.

A model abstracts details relevant for analysis. Analyses rarely discuss particular policies; they usually focus on **specific characteristics** of policies, because many policies exhibit these characteristics; and the more policies with those characteristics, the more useful the analysis. By the HRU result (see Theorem 3–2), no single nontrivial analysis can cover all policies, but restricting the class of security policies sufficiently allows meaningful analysis of that class of policies.