Access Control Systems

Security, Identity Management and Trust Models

Access Control Systems

Security, Identity Management and Trust Models

by

Messaoud Benantar
IBM Corp, Austin, TX, USA

Springer

Dr. Messaoud Benantar
IBM Corp.
Austin, TX, USA

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

Access Control Systems:
Security, Identity Management and Trust Models
by Messaoud Benantar

ISBN-10: 0-387-00445-9 e-ISBN-10: 0-387-27716-1
ISBN-13: 978-0-387-00445-7 e-ISBN-13: 978-0-387-27716-5

Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 10911600, 11525653

springeronline.com

“To my little world – Elyes, Aicha and Houda. To my elementary school teachers – Abdelmadjid Bouanane and Messaoud Berrou.”

Preface

Secure identification of users, programming agents, hosts, and networking devices is considered the core element of computing security. Rarely is anonymity a desired goal of systems, networks, and applications. This aspect is dictated largely by the extent in which computing has evolved to automate many facets of critical human activities, such as in businesses and even in processes that can have direct effects on human lives. To that end every unit of computing in modern systems with a relative level of security is attached to an authenticated identity associated with it. This enables deterministic accountability and lays the foundation for responsible and secure computing, as we present in chapter 1. We emphasize the major aspects relating to identification and access control and define the basic concepts that collectively form the foundation for computing security.

An identity in computing reflects real-life entities in that its level of granularity can be coarse (such as representing an organization; a group of people) or can represent a specific individual or a particular computing device. The premise of achieving deterministic accountability is centered on the processes that support coherent and consistent identity management where a one-to-one correspondence of an identity to a real entity, its owner, can be achieved. Assurance in identity, referred to as *identity trust*, is established through authentication. In computing security trust is computable. The authentication process is based on providing what is called the *proof of identity possession*, while uniqueness of an identity is generally parameterized by referencing a well defined naming space. The latter can be as simple as a local registry of a centralized system or as wide and global as the Internet. The level of trust in an identity varies depending on the proof presented to establish it. Although trust in computing spans all elements that contribute to enforcing system and networking controls including the integrity of identity repositories and that of governing policies, evidently it is all predicated on the trust that a system or a network establishes in an identity.

The Evolution of computing—from centralized to distributed systems and now well into the global era of the Internet—has tremendously increased the complexity associated with identity management and trust. Chapter 2 introduces the reader to the elements of identity management.

We provide a taxonomy of various existing schemes based on the defining scope of an identity and discuss the benefits and limitations of each. We present the elements of federated identity and show how identity has moved from being simply a concept and a manipulated data construct that has little effect on processing to becoming, by its own right, the object of systems management in what is known as *identity provisioning*.

The simplistic view in the centralized computing era is characterized by the scope of identity being limited to a locally managed user registry. The naming space from which an identity is drawn is generally flat and implicitly qualified by the computer system in which it is defined. It ceases to exist uniquely outside this limited boundary. Since the proof of identity possession remained in the confines of an organization's computing infrastructure, it largely relied on the use of passwords.

The network-computing era raised the scope of an identity to the network level, thereby becoming visible to all computing systems attached to a network. It pushed the limits into network wide identity registries and authentication protocols that are based on various encryption schemes, most notably secret key. When multiple registries are used, consistency and synchronization of identity attributes became a necessity. This era also highlighted the need for network wide single sign-on and presented eloquent solutions to it. The network wide scope increased the functional requirements needed for securely establishing and maintaining trust in an identity. The network-security context came into existence to represent this trust.

The era of Internet computing is seeing an unprecedented need for reliable identity-management and trust mechanisms. Conducting business transactions over public networks requires secure processes for establishing a security context before it is attached to a particular transaction, verifying it, propagating it from end point to end point, and managing its life cycle. The multitude of Web services that can potentially collaborate behind the scene of a single-end-user transaction requires secure propagation of identity trust and interoperable models of profiling attributes. Several models of trust propagation have emerged.

The Web model of computing necessitates a Web model of identity management. Identity attributes, known as *profiles*, need to be consistently interpreted and exchanged across organization boundaries in arbitrary ways. Profiles that are associated with the same entity may need to maintain a mapping to each other and be kept synchronized. Privacy concerns have emerged to an extent never seen before. Remedies to these issues need to apply to every level of profile attributes from coarse to finer components and should be based on individual concerns, organizational policies, and emerging standards. To facilitate and ease collaboration, organizations may find the need to be federated together to form entities whose boundaries are seamless to users. The transparency provided by these federations allows entities to undergo a single registration process and experience the benefits of single sign-on throughout a virtually larger organization.

Chapter 3 is concerned with the elements of identity trust. We survey existing models as they relate to assurance in an identity. We begin with the simplistic method of sharing secrets and subsequently delve into the public-key aspect of trust. Various public-key-based trust models are presented. Identity-management processes alone are not sufficient if they are not coupled with a strong foundation of trust, particularly across organizations. The ultimate need for the secure establishment of an identity is to impose controls over the entitlements, which can be granted or denied to the associated entity. The goal is to base access-control decisions on secure foundations.

Trust in an identity and its associated profile attributes is generally intended as a prerequisite for a secure determination of entitlements. Access control is founded on the establishment of secure identity contexts. Assurance in that foundation is a key element in secure access-control implementations. Other important aspects include the processes enabling access decision making and the adoption of access policies that are based on well-defined models. Management of access-control supporting constructs (such as policy maintenance) and of the provisioning of entitlements to various entities is also an important element. Subsequent to the initial introduction of existing paradigms of information access control in Chapter 1, we discuss the details of the mandatory-access-control (MAC) model in Chapter 4. We demonstrate the ease of information-flow analysis in this model and present a few of its variants. In Chapter 5 we delve into the access-matrix model and focus on all aspects of discretionary access control (DAC). We introduce the reader to the elements of safety and show the complexity of analyzing access-control systems in a generalized form. Chapters 6 and 7 present the take-grant and the schematic models, respectively. These schemes are of lesser generality than the access-matrix model but have computable safety properties. Chapter 8 presents the details of role-based access control (RBAC) beginning with the basic concepts to the complex aspects of mapping DAC and MAC onto RBAC. Information-flow analysis of RBAC is discussed and the RBAC standard is highlighted.

Models help elevate access-control management to a level that is concise and in some cases even formal. Modeling is an important tool for attempting to define the bounds of information flow in any given computing environment. Access-control models follow along existing paradigms of information flow. Two major such paradigms are known to date, discretionary and mandatory. *Discretionary access control* empowers resource owners in divulging access to others. The flexibility of this paradigm, however, removes any possibility for defining the limits that can be reached by a given protection state. Such states are unbounded, and the flow of information is generally unpredictable. Nevertheless, DAC is the most widely adopted access-control paradigm. It naturally fits many of real-life processes that govern access to resources based on ownership.

Mandatory access control leaves no powers to end entities in deciding the flow of information. Instead, select administrators of an organization grant or deny access by assigning security classifications to resources and active entities (such as computing devices and programming subsystems and users, referred to as labels and clearances, respectively). Access decisions are then made in accordance with a partially ordered relationship between labels and clearances in what is known as *dominance* or the lack thereof. Contrary to DAC, dissemination of information in MAC is predictable as it follows a lattice structure that accurately determines the bounds of information flow. MAC lends itself well to military environments, while it is generally regarded as a handicapping measure in commercial environments.

Role-based access control has emerged in recent years as a generalized access model that although it encapsulates more of discretionary flavor than mandatory, it theoretically applies to DAC as well as MAC policies. RBAC seems to fit naturally into modeling access control. Its main advantage is in the simplification of management and administrative tasks of governing security policies. Additionally, it lends itself well to the *separation-of-duty* (SoD) principle. SoD can be viewed in many respects as a bridge between DAC and MAC policies. Like in MAC, the administrative tasks play an important role in how information is disseminated in RBAC. Like DAC, RBAC is capable of maintaining the concept of resource ownership.

Although the elements surrounding RBAC are interpreted with relative uniformity across the computing industry, interoperability of implementations remains elusive. The absence of common-role semantics and unified policy representations makes it difficult to switch from one environment to another. Nevertheless, a recent attempt by the National Institute of Standards and Technology (NIST) at standardizing some of the RBAC aspects can be an important step forward. We devote chapter 8 to this important topic.

This book is a modest attempt at discussing these elements of computing security. I hope you find it enjoyable to read and that it clarifies these concepts for you.

Messaoud Benantar
Austin, Texas, USA

Contents

Chapter 1

Foundations of Security and Access Control in Computing

Introduction

Access control in computing is motivated by the need to divulge access to information and available computing resources and services to authorized entities only. An *entity* is a generic term that refers to an active agent capable of initiating or performing a computation of some sort (for example, an end user invoking a command or a program, a programming agent acting on behalf of a user, a running daemon process, a thread of execution, a hosting system, or a networking device). Access modes can be broadly categorized into the ability to read or write information whether in the address space of an executing process, on a secondary storage, or on a network or a peripheral device. This ability can be explicitly expressed by a direct privilege possessed by the acting entity or indirectly through services and computing tasks that the entity is allowed to execute. A purist may pose the question of whether temporarily modifying computer information without having to read it and in a way that leaves its final state unchanged is consistent with the definition of access control. The likely answer is that such activity constitutes a breach to access control and thus it should be guarded against. Otherwise, one of the fundamental security tenets of resource availability becomes at risk of being compromised. Availability of computing resources has indeed stood as a system and network security concern of its own. Furthermore, concurrent access to information that is being modified even temporarily by authorized or unauthorized entities is clearly unacceptable.

Evolution of computing systems from single-user to multiuser machines led to the necessity of shielding users and running processes from one another. Early protection mechanisms consisted of hardware and operating systems components. Subsequently, policy-based authorization subsystems have emerged. Controlling access to computing systems is the first defense against disclosing information to unauthorized entities. Systems and network access is based on trusted methods for identifying users and programming agents. Secure identification is the cornerstone of modern computing security. The advent of networking and distributed computing has led to the proliferation of computing identities. Consequently, identity management has evolved as a discipline of its own. The goal is to mitigate the cost of maintaining identity repositories that may exist in the potentially a myriad of systems used by a single enterprise, enforcing consistency and achieving unambiguous mapping of identities representing the same entity or multiple entities collaborating together. Automation of interenterprise exchanges has further necessitated the drive for federated identity systems. As a result, the scope of an identity is extending well beyond the confines of an organization. With all the associated complexities, a purist perspective seeks a unified model of secure identification. Although this is far from being achieved in the real world, any such attempts can only benefit computing security.

Real-world examples of access control are abundant and vary according to the needs and policies dictated by the circumstances. At a basic level, users of the same organization are granted access to shared computing resources based on the roles each user is entitled to within the organization. An enterprise may be concerned over losing its competitive edge should its trade secrets become known to its competitors. A financial institution has every need to confine updates in its records to legitimate transactions only and to protect them from exposure to unauthorized individuals and institutions. While a patient's medical records may not be of any immediate financial gain, one cannot put a price to their privacy.

Access control is evolving from its traditional host-centric paradigm to resources and entities that transact over large networks as wide as the Internet. The low-level access-control privileges of the basic read and write of information are now moving up a level higher to include attributes that make up a profile for an entity. These are the elements that mimic real-life user entitlements such as the privilege of having a banking account, having a credit-card number, or being assigned a well-defined role. The processes needed to maintain entity profiling gave rise to what is referred to as identity management, which is indeed a prelude to any access-control mechanism. It is concerned with the trusted methods of managing and exchanging entity entitlements on various computing systems and resource managers. Identity management forms the foundation on which access control is based.

In this chapter we introduce the main concepts behind computing security. We begin with a brief overview of security threats. We then elaborate on the major elements of systems security, in particular the aspects surrounding identification and authentication. We highlight the importance of system integrity as a prelude to secure computing. We define what is meant by a security context and discuss its propagation along the units of computing work. Subsequently, we delve into the paradigms of access control and outline the elements surrounding trust and assurance, including an introduction to the confinement problem. We conclude with an overview of the major security-design principles.

Elements of Systems Security

A threat by definition is a situation in which any protection mechanisms that govern access to a computing system may become subject to harm. Such protection mechanisms are driven by what is called a *security policy*. We discuss the concept of a security policy in further detail later in the chapter. Security threats are analogous to harmful activities that are bound to happen and thus convey the meaning of a pending attack. The latter makes the threat a reality. Threats are made possible due to *vulnerabilities*, also referred to as *weaknesses*, either in the mechanisms enforcing a particular security policy or in the operational controls of that policy (such as those having to do with configuration parameters). Mechanism-related vulnerabilities can be due to design or implementation flaws. Dormant vulnerabilities represent a risk. A *risk* is a measure of potential harm that can be realized when a threat is executed. Some of the known categories of security threats include identity theft through masquerading or spoofing, unauthorized access to resources, unauthorized disclosure or modification of data, and denial of service attacks.

Security in computing can be viewed as having the following elements:

- Secure entity identification, known as *authentication* and which we refer to as *identity establishment*;
- Confining actions of an established identity to its designated entitlements for services and computing resources, known as *resource access control*;
- Data integrity, confidentiality, and origin authenticity, broadly referred to as *data and message security*;
- Prevention from denial of taking part in a transaction, whether as an initiating or a receiving party, known as *nonrepudiation*;
- Resource *availability* to thwart against the denial of service attacks.

The fundamental prerequisite for the integrity and soundness of any access-control or other security mechanisms is the secure establishment of identities. For example, the lack of enforcement for secure establishment of identities, makes all attempts to enforce an access policy virtually useless.

Identity Establishment

Identity establishment is concerned with the methods by which a user, a running process, or a thread of execution is securely associated with a legitimate entity. Recall that an entity may represent a single user, a group of users, an entire organization, a host system, or some networking device. Establishing an identity is the means of concluding that indeed the identity in use corresponds to the entity that it claims to be and thus is said to be authentic. *Authentication* is the secure identification of entities in which a proof of possessing an identity is verified. An entity's access to a system is encapsulated in what has become known as an *account*. Engaging in an act of authentication can take place on every attempt to access a controlled computing system, known as a *login*, when a service from an application is requested, or each time a network access is performed. Varying system and network security policies as well as application requirements can dictate the frequency of entity authentication.

The evidence resulting from an established identity is maintained by the computing device in what is referred to as a *security context*. The latter remains securely attached to every unit of work requested by the corresponding entity. A security context can be exchanged locally across address spaces and may be transmitted over a network embodied in the request with which it is associated.

Resource Access Control

Access control, one of the central themes of this book, is also referred to as *access authorization* or simply *authorization*. It is about enforcing a predefined access policy. The goal is to confine the actions of an entity only to the services and to the computing resources that it is entitled to. To prevent an access policy from subversion, the controls that enforce it should be foremost capable of binding computing activities to authenticated identities at any fine level of computation, the scope of which may be an entire address space or at the task and thread level. These bindings are known as *secure associations*. A safe access-control policy prevents leakage of access to unauthorized users directly or indirectly in any state of the underlying computing system. As we have already mentioned, identity establishment is the cornerstone of enforcing any resource access-control policy.

Data and Message Security

Although the term *data security* is generic, its use is mainly concerned with modification detection, origin authenticity, and confidentiality of data that is being processed in-memory, or while residing on a storage medium or during transmission over a computer network (i.e., a message). Modification detection or simply data integrity alone is not of value to data security unless it is combined with origin authenticity. An eavesdropping entity may apply the same data-integrity procedures after having intercepted and modified data items, leading the receiving entity to successfully verify the integrity of the breached data but without realizing it was modified. Thus, data integrity is usually combined with some form of origin authenticity, ensuring that an integrity-check sum is indeed generated by a legitimate entity, the original source of the data. Secure data integrity, one combined with origin authenticity, protects against an unauthorized update of data.

Confidentiality is the process of sealing data using a keyed data-scrambling algorithm so that only a designated entity, one with knowledge of the key, is able to apply the reverse transformation and retrieve the data in its original form. The goal is to prevent disclosure of information to unauthorized entities. In a sense, data confidentiality can be used as a mechanism for enforcing access to information. The underlying cost, however, can be prohibitive so that access-control mechanisms are generally not based on data confidentiality. Data confidentiality remains a discipline of its own in security. It is selectively applied to sensitive information that when disclosed results in measurable or un-measurable loss of some kind.

Nonrepudiation

Nonrepudiation of action is the process by which an entity is prevented from denying participation in a transaction either as an initiating/sending or a receiving end. The definition is ultimately applicable to preventing any process or a thread of execution running on behalf of an end user to circumvent the binding of the acting identity with the legitimate entity. Although one might argue that nonrepudiation can be accomplished simply by producing audit and transaction trails in a secure and a controllable fashion, a purist would assert that a legally binding nonrepudiation can be very hard to realize. Denial may always take one form or another. Nevertheless, digital signatures based on public key cryptography and a combination of tamper-proof hardware and software modules have come a long way toward establishing verifiable nonrepudiation services, particularly for initiating entities (i.e., those generating information).

Availability

Availability addresses the issue of disrupting access to computing resources and services. The type of disruption may range from compromising the functions of a particular service or a system to completely denying access to it. Under all circumstances, it is natural for users of any computing service to expect reasonable response times that are comparable to or much better than human-to-human interactions (over a telephone line, for instance) to attain the same service.

Protecting computing resources from extreme degradation of performance or from deliberate denial of service takes priority over the enforcement of any access-control policy. A denial-of-service (DOS) attack is one in which a deliberate high volume of bogus requests are sent to a service provider. The intent is to keep legitimate users of the service from using it. An attack as such may bring the service to its threshold capacity, leaving it dedicated to handling malicious requests instead of legitimate ones. The manifestation may result in extremely slow response times and potentially may lead to a complete inhibition of service and ultimately a shutdown due to the exhaustion of runtime resources, such as real or secondary storage or network sockets. Powerful attacks as such may further bring down an entire network as wide as the Internet to a crawl.

When authorized users are not able to send requests or reach a service, it becomes a secondary concern to have that service enforce an access-control policy. Furthermore, the mere existence of the service is entirely threatened. Security mechanisms that protect the availability of computing resources guard against various threats of interruption and deliberate actions of slowing down a service or rendering it completely inaccessible. Detection and prevention of DOS attacks have emerged as among the leading security issues in this era of computing over public networks.

It should be noted that disruptions leading to denial of service may occur at different locations along the path between a client and a server, including the following:

- *In the environment of the service* Here the service is prevented from obtaining resources needed for its proper execution. The attacker focuses on exhausting computing resources of the system in which the service is hosted.
- *In the environment of the client* The target service is diverted from responding to legitimate requesters and dealing with useful communications by way of attempting to respond to a massive bombardment of random client messages instead.
- *Along the path between clients and the server* The attacker intercepts and then discards useful requests to the service.

Cost of Security

Security in computing, as in anything else, comes with cost and overhead. That cost should be put in perspective with the value of the protected resources. The cost of security has to be proportionate to the losses incurred from any security breaches. Insignificant losses do not require significantly higher security costs. Measuring potential loss is not a deterministic process; worst-case scenarios therefore are to be assumed. In quantifiable terms, the cost of security should be less than that of entirely replacing a protected computing asset including its data and functionality. Being able to quantify various elements of risk enables the development of informed policies that balance the cost of security with the benefits of increased safety. Threats have to be considered even in highly secure environments. The probability of ruin in a computing infrastructure, even when relatively low, should be the driving factor behind the provision of security. However, one cannot always put cost to security. Invasion of privacy (such as publicly exposing a person's medical records) can be detrimental to the person, even when seemingly no quantifiable physical harm is inflicted on the person and the health-care provider.

System Integrity: A Prelude to Security

Integrity of information processing was the focus of attention in early stages of the developments in information technology (IT). First, the need for a strict separation between a running control program and user or application programs was addressed even in basic single-user systems. Operating systems and hardware advancements such as those pioneered by the IBM System/360 and System/370 family have led to multiuser systems that accommodate a large number of users. The execution of multiple processes addressing a common memory meant that one process must be prevented from overwriting memory locations that are assigned to another process. Address-space separation, therefore, had to be maintained in both the virtual storage assigned to a process and the real memory blocks used at runtime. In early IBM systems this problem was addressed with storage-protection keys where a particular process and the storage assigned to it are associated with a unique storage key that must match if the process is allowed to access the storage. Any attempt by a process to store data outside of its assigned blocks of memory is recognized by the hardware due to mismatched storage-protection keys.

In IBM's System/360 through System/390 and beyond, the control program defining the operating system is isolated from user programs by means of a two-state instruction execution environment. These two states are called *supervisor state* and *problem-program state*. A special set of machine instructions, including input/output (I/O) commands to the I/O channels and memory as well as address-space-management instructions are operable only when the system is running in supervisor state. The control program typically executes in supervisor state while user programs always execute in the problem-program state. When an application requests the services of the control program (such as performing I/O), a request is issued to the control program. The control program, executing in the supervisor state, first examines the request to make sure that it will not exceed the logical boundaries of the problem program before the request is executed.

The assurance provided by modern operating systems in isolating concurrently running user applications and control programs is the key to enforcing the security controls that a computer system provides. Such isolation is further extended to finer levels of computing units—that of execution threads. The needs for isolation equally apply to the threads executing in a single address space. Figure 1.1 illustrates the concept of isolation across operating system and user processes as well as threads. A classical example of the benefits from well-designed isolation mechanisms are found in the features that are embedded in the control program of the System/390 and its derivative platforms. These mechanisms are extended to cover new software components that are tightly related to the control program. One of these components is the security service layer, which is invoked by various resource managers and also by system components to mediate access to system resources.

Trusted Computing Base

A *trusted computing base* (TCB) is defined as the totality of protection mechanisms within a computer system, including hardware, firmware, and software, the combination of which is responsible for enforcing a security policy

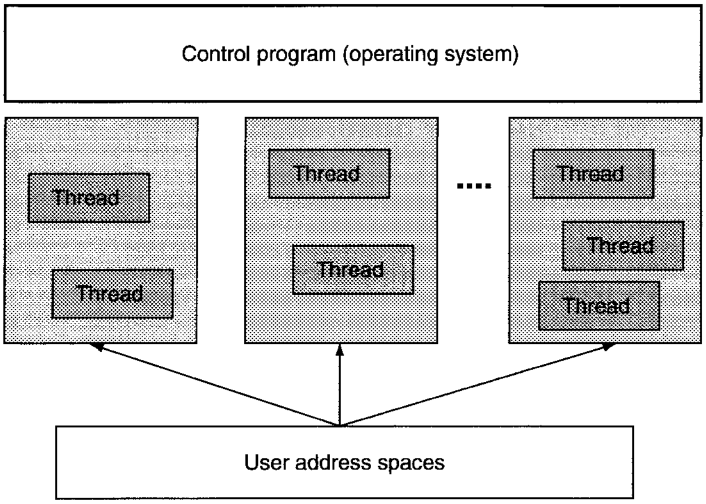

FIGURE 1.1 Isolation of program execution in modern operating systems

[ABRA95]. The ability of a trusted computing base to enforce a security policy correctly depends foremost on the integrity, correctness, and protection of the mechanisms implementing the elements of the TCB itself. Similarly, a *network trusted computing base* (NTCB) is defined as the totality of protection mechanisms within a network including hardware, firmware, and software, the combination of which is responsible for enforcing a networkwide security policy. A *mechanism* is the term used to refer to a specific paradigm, model, or a construct that is used in the implementation of a particular service. A security service enforcing a policy is therefore a combination of security mechanisms. Trust in a TCB means the components and mechanisms implementing the enforcement of controls dictated by a security policy behave in an expected manner. The expectation here is that the TCB should not subvert the policy that it is designed to enforce. Basic to the element of trust in the TCB is its correctness and overall system integrity.

The general method of defining the boundaries of a TCB is that any software, firmware, or a hardware component that has the ability to subvert a security policy is considered to be part of an applicable TCB or NTCB. Breaching a TCB is usually accomplished by carrying an attack that the designer of the TCB had not anticipated. Building an ideal TCB, therefore, requires exhausting all possible attacks. While it may seem that the elements of network TCB are scattered and disjoint, in practice trust is a continuous concept throughout that follows the information flow. Applicable trust properties should remain invariant when information is residing on a storage system, within a thread of execution, during an exchange of data across address spaces, or while in transmission over a network.

Users, Principals, Subjects, and Objects

The term *user* in computing has been traditionally equated with a human being. Its use conveys a unique association between a computing system and an entity that can be a human being or some programmable agent. User information is generally encapsulated in an *account*, sometimes referred to as a *profile*. A user account contains information about authentication as well as authorization credentials and may contain a set of attributes describing the user (such as a name, a serial number, an organization name, and so forth). Each user account is associated with an identifier that must be unique in the naming space of the underlying computing system.

While a user represents an entity external to a computing system, a *principal* generally refers to an entity's internal representation to a computing system. Each user may have several principals associated with it. Each principal, on the other hand, is associated with one user only. The principal construct defines the runtime association between a computing task and a particular user and generally encapsulates a subset of the entitlements of that user. The scope of entitlement is dependent on the application to which the user signs in. For instance, besides being an employee of Zeta, Inc., user Aicha is participating in two projects within her company codenamed Green and Blue. Each of these projects requires special privileges. In the absence of a dynamic policy that constraints the entitlements of an entity based on its role, Aicha may be assigned three principal identities, all of which point to the same user. The first is Aicha, being the basic identity in the system; AichaB and AichaG correspond to projects Blue and Green, respectively. The relationship of the secondary identities AichaB and AichaG to the main identity Aicha should be well maintained in the system to establish an accurate binding between a physical entity, such as a user and all of its principal identities. A profile representing the primary identity of a user should point to all principal identities associated with that user.

A *subject* is the term used to identify a running process, a program in execution. Each subject assumes the identity and the privileges of a single principal. A principal may launch several processes within a single login session and thus will be associated with multiple subjects, each of which inherits the identity of the login session. Figure 1.2 illustrates the relationships between a user, a principal, and a subject.

An *object* generally refers to a passive entity (i.e., one that is an information receptacle such as a file, or a record in a database). An object, however, may indicate an active device from the system's resource pool (such as a network printer, or further can be a programmable service that is managed as a resource).

It is worth noting that in many cases we simply encounter the basic scenario in the relationships among a user, principal and subject where the user, the principal, and the subject are all the same. In the security literature the term *principal* is generally used to mean an active entity that is capable of