Unit 9: Security

Contents:

9.1 Introduction to Security

Security in distributed systems can roughly be divided into two parts.

One part concerns the **communication between users or processes**, possibly residing on different machines. The principal mechanism for ensuring secure communication is that of a secure channel. Secure channels, and more specifically, authentication, message integrity, and confidentiality, are discussed in a separate section.

The other part concerns **authorization**, which deals with ensuring that a process gets only those access rights to the resources in a distributed system it is entitled to. Authorization is covered in a separate section dealing with access control. In addition to traditional access control mechanisms, we also focus on access control when we have to deal with mobile code such as agents.

It is also important in naming, and pay attention to the rather nasty problem of making sure that the name used to retrieve an object belongs to that object, but also how to combine secure naming with human-friendly names.

Secure channels and access control require mechanisms to distribute cryptographic keys, but also mechanisms to add and remove users from a system.

Security threats, policies, and mechanisms

Security in a computer system is strongly related to the notion of **dependability**. Informally, a dependable computer system is one that we justifiably trust to deliver its services.

Dependability includes

- availability,
- reliability,
- safety, and
- maintainability

However, if we are to put our trust in a computer system, then **confidentiality** and **integrity** should also be taken into account.

Confidentiality refers to the property of a computer system whereby its information is disclosed only to authorized parties.

Integrity is the characteristic that alterations to a system's assets can be made only in an authorized way. In other words, improper alterations in a secure computer system should be detectable and recoverable. Major assets of any computer system are its hardware, software, and data.

Another way of looking at security in computer systems is that we attempt to protect the services and data it offers against security threats. There are four types of security threats to consider:

1. **Interception**: The concept of interception refers to the situation that an unauthorized party has gained access to a service or data. A typical example of interception is where communication between two parties has been overheard by someone else. Interception also happens when data are illegally copied, for example, after breaking into a person's private directory in a file system.

2. **Interruption**: An example of interruption is when a file is corrupted or lost. More generally interruption refers to the situation in which services or data become unavailable, unusable, destroyed, and so on. In this sense, denial of service attacks by which someone maliciously attempts to make a service inaccessible to other parties is a security threat that classifies as interruption.

3. **Modification**: Modifications involve unauthorized changing of data or tampering with a service so that it no longer adheres to its original specifications. Examples of modifications include intercepting and subsequently changing transmitted data, tampering with database entries, and changing a program so that it secretly logs the activities of its user.

4. **Fabrication**: Fabrication refers to the situation in which additional data or activity are generated that would normally not exist. For example, an intruder may attempt to add an entry into a password file or database. Likewise, it is sometimes possible to break into a system by replaying previously sent messages.

Simply stating that a system should be able to protect itself against all possible security threats is not the way to actually build a secure system. What is first needed is a description of security requirements, that is, a **security policy**.

A security policy describes precisely which actions the entities in a system are allowed to take and which ones are prohibited. Entities include users, services, data, machines, and so on. Once a security policy has been laid down, it becomes possible to concentrate on the security mechanisms by which a policy can be enforced. Important security mechanisms are:

- **Encryption**: Encryption is fundamental to computer security. Encryption transforms data into something an attacker cannot understand. In other words, encryption provides a means to implement data confidentiality. In addition, encryption allows us to check whether data have been modified. It thus also provides support for integrity checks.
- **Authentication**: Authentication is used to verify the claimed identity of a user, client, server, host, or other entity. In the case of clients, the basic premise is that before a service starts to perform any work on behalf of a client, the service must learn the client's identity (unless the service is available to all). Typically, users are authenticated by means of passwords, but there are many other ways to authenticate clients.
- **Authorization**: After a client has been authenticated, it is necessary to check whether that client is authorized to perform the action requested. Access to records in a medical database is a typical example. Depending on who accesses the database, permission may be granted to read records, to modify certain fields in a record, or to add or remove a record.

- **Auditing**: Auditing tools are used to trace which clients accessed what, and in which way. Although auditing does not really provide any protection against security threats, audit logs can be extremely useful for the analysis of a security breach, and subsequently taking measures against intruders. For this reason, attackers are generally keen not to leave any traces that could eventually lead to exposing their identity. In this sense, logging accesses makes attacking sometimes a riskier business.

Design issues

A distributed system, or any computer system for that matter, must provide security services by which a wide range of security policies can be implemented. There are a number of important design issues that need to be taken into account when implementing general-purpose security services.

Some of these issues are: **focus of control**, **layering of security mechanisms**, and **simplicity**.

Focus of control: When considering the protection of a (possibly distributed) application, there are essentially three different approaches that can be followed, as shown in Figure.

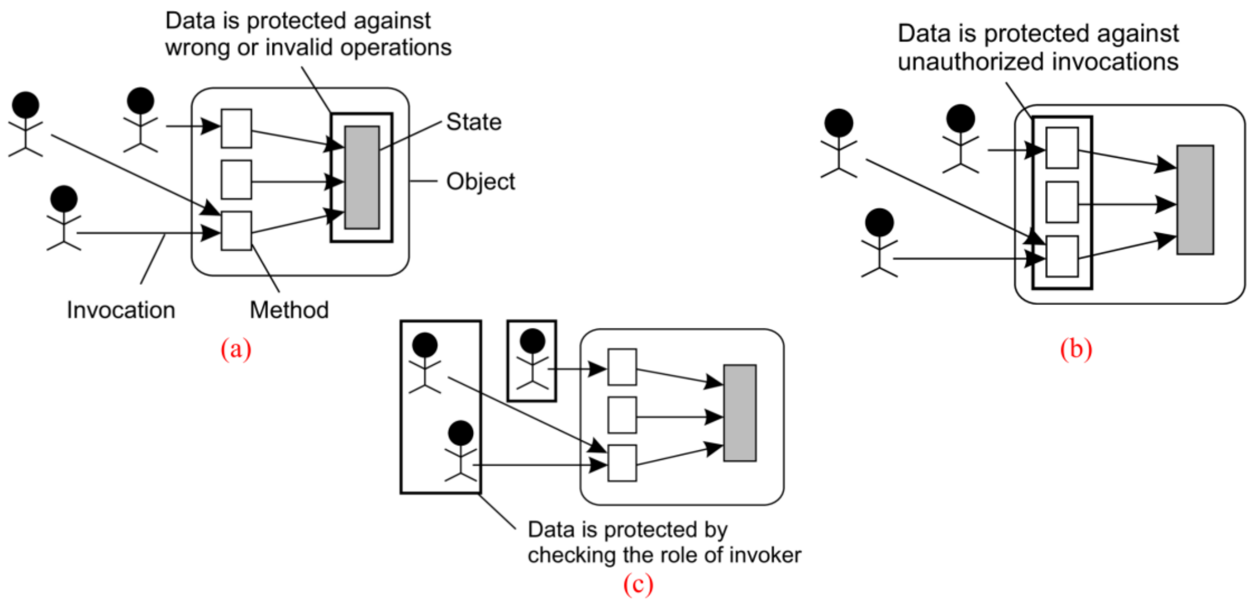

Data is protected against wrong or invalid operations
State
Object
Invocation
Method
(a)
Data is protected against unauthorized invocations
(b)
Data is protected by checking the role of invoker
(c)

Figure 9.1: Three approaches for protection against security threats. (a) Protection against invalid operations (b) Protection against unauthorized invocations. (c) Protection against unauthorized users.

The first approach is to concentrate directly on the protection of the data that is associated with the application. By direct, we mean that irrespective of the various operations that can possibly be performed on a data item, the primary concern is to ensure data integrity. Typically, this type of protection occurs in database systems in which various integrity constraints can be formulated that are automatically checked each time a data item is modified.

The second approach is to concentrate on protection by specifying exactly which operations may be invoked, and by whom, when certain data or resources are to be accessed. In this case, the focus of control is strongly related to access control mechanisms. For example, in an object-based system, it may be decided to specify for each method that is made available to clients which clients are permitted to invoke that method. Alternatively, access control methods can be applied to an entire interface offered by an object, or to the entire object itself. This approach thus allows for various granularities of access control.

A third approach is to focus directly on users by taking measures by which only specific people have access to the application, irrespective of the operations they want to carry out. For example, a database in a bank may be protected by denying access to anyone except the bank's upper management and people specifically authorized to access it. As another example, in many universities, certain data and applications are restricted to be used by faculty and staff members only, whereas access by students is not allowed. In effect, control is focused on defining roles that users have, and once a user's role has been verified, access to a resource is either granted or denied. As part of designing a secure system, it is thus necessary to define roles that people may have, and provide mechanisms to support role-based access control.

Layering of security mechanisms

An important issue in designing secure systems is to decide at which level security mechanisms should be placed. A level in this context is related to the logical organization of a system into a number of layers. For example, computer networks are often organized into layers following some reference model.

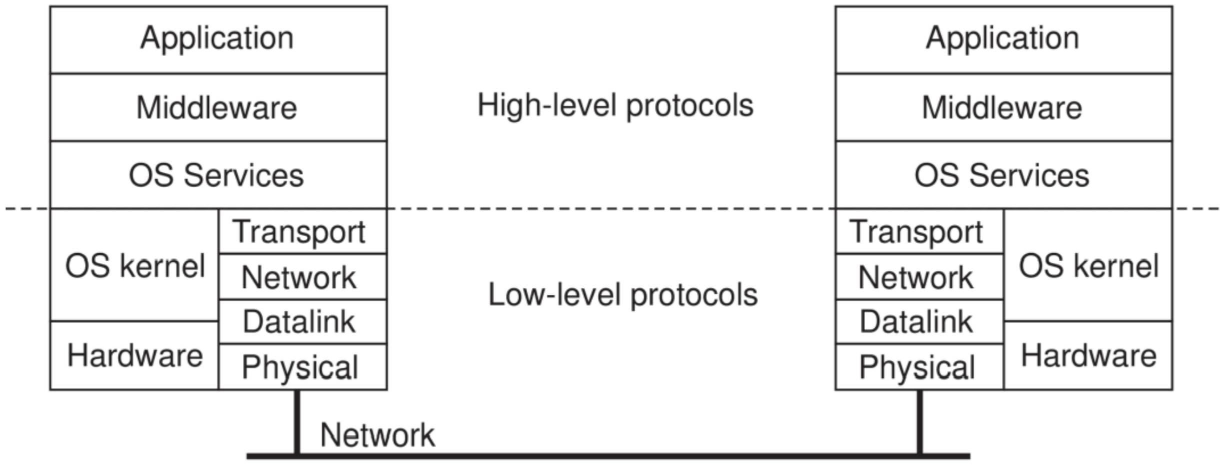


Figure separates general-purpose services from communication services. This separation is important for understanding the layering of security in distributed systems and, in particular, the notion of trust. The difference between trust and security is important. A system is either secure or it is not (taking various probabilistic measures into account), but whether a client considers a system to be secure is a matter of trust. Security is technical; trust is emotional. In which layer security mechanisms are placed depends on the trust a client has in how secure the services are in a particular layer.

Simplicity

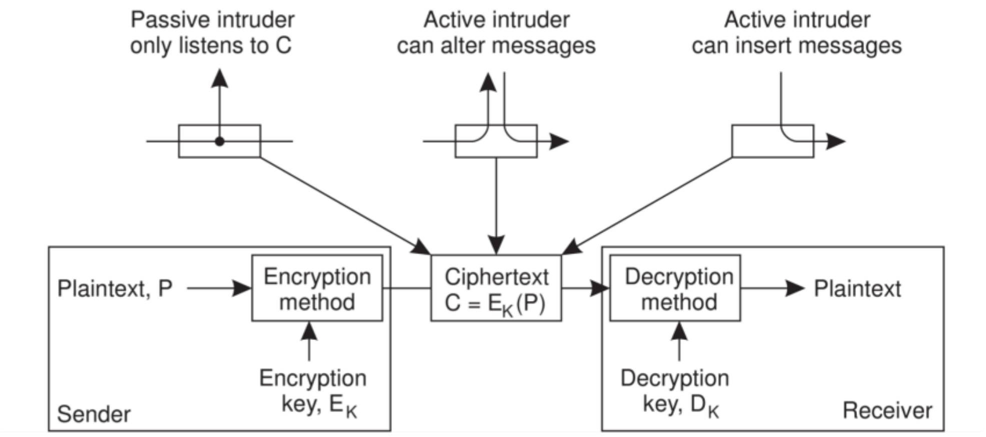
Another important design issue related to deciding in which layer to place security mechanisms is that of simplicity. Designing a secure computer system is generally considered a difficult task. Consequently, if a system designer can use a few, simple mechanisms that are easily understood and trusted to work, the better it is. Unfortunately, simple mechanisms are not always sufficient for implementing security policies.

The application itself is inherently complex and introducing security only makes matters worse. An example application domain involving complex security protocols is that of digital payment systems. The complexity of digital payment protocols is often caused by the fact that multiple parties need to communicate to make a payment. In these cases, it is important that the underlying mechanisms that are used to implement the protocols are relatively simple and easy to understand. Simplicity will contribute to the trust that end users will put into the application and, more importantly, will contribute to convincing the designers that the system has no security holes.

Cryptography

Fundamental to security in distributed systems is the use of cryptographic techniques. The basic idea of applying these techniques is simple. Consider a sender S wanting to transmit message m to a receiver R. To protect the message against security threats, the sender first encrypts it into an unintelligible message m’, and subsequently sends m’ to R. R, in turn, must decrypt the received message into its original form m.

Encryption and decryption are accomplished by using cryptographic methods parameterized by keys, as shown in figure below. The original form of the message that is sent is called the plaintext, shown as P in Figure the encrypted form is referred to as the ciphertext, illustrated as C.



While transferring a message as ciphertext C, there are three different attacks that we need to protect against, and for which encryption helps.

First, an intruder may intercept the message without either the sender or receiver being aware that eavesdropping is happening. Of course, if the transmitted message has been encrypted in such a way that it cannot be easily decrypted without having the proper key, interception is useless: the intruder will see only unintelligible data. (By the way, the fact alone that a message is being transmitted may sometimes be enough for an intruder to draw conclusions. For example, if during a world crisis the amount of traffic into the White House suddenly drops dramatically while the amount of traffic going into a certain mountain in Colorado increases by the same amount, there may be useful information in knowing that.)

The second type of attack that needs to be dealt with is that of modifying the message. Modifying plaintext is easy; modifying ciphertext that has been properly encrypted is much more difficult because the intruder will first have to decrypt the message before he can meaningfully modify it. In addition, he will also have to properly encrypt it again or otherwise the receiver may notice that the message has been tampered with.

The third type of attack is when an intruder inserts encrypted messages into the communication system, attempting to make R believe these messages came from S. Again, encryption can help protect against such attacks. Note that if an intruder can modify messages, he can also insert messages.

There is a fundamental distinction between different cryptographic systems, based on whether or not the encryption and decryption key are the same.

In a **symmetric cryptosystem**, the same key is used to encrypt and decrypt a message: $P = D_K(E_K(P))$

Symmetric cryptosystems are also referred to as secret-key or shared-key systems, because the sender and receiver are required to share the same key, and to ensure that protection works, this shared key must be kept secret; no one else is allowed to see the key. We will use the notation $K_{A,B}$ to denote a key shared by A and B.

In an **asymmetric cryptosystem**, the keys for encryption and decryption are different, but together form a unique pair. In other words, there is a separate key KE for encryption and one for decryption, K_D, such that $P = D_{KD} (E_{KE} (P))$

One of the keys in an asymmetric cryptosystem is kept private; the other is made public. For this reason, asymmetric cryptosystems are also referred to as public-key systems. In what follows, we use the notation K^+_A to denote a public key belonging to A, and K^-_A as its corresponding private key.

9.2 Secure Channels

The client-server model is a convenient way to organize a distributed system. In this model, servers may possibly be distributed and replicated, but also act as clients with respect to other servers. When considering security in distributed systems, it is once again useful to think in terms of clients and servers. In particular, making a distributed system secure essentially boils down to two predominant issues.

The first issue is how to make the communication between clients and servers secure. Secure communication requires authentication of the communicating parties. In many cases it also requires ensuring message integrity and possibly confidentiality as well. As part of this problem, we also need to consider protecting the communication within a group of servers.

The second issue is that of authorization: once a server has accepted a request from a client, how can it find out whether that client is authorized to have that request carried out? Authorization is related to the problem of controlling access to resources.

The issue of protecting communication between clients and servers, can be thought of in terms of setting up a secure channel between communicating parties. A secure channel protects senders and receivers against interception, modification, and fabrication of messages. It does not also necessarily protect against interruption. Protecting messages against interception is done by ensuring confidentiality: the secure channel ensures that its messages cannot be eavesdropped by intruders. Protecting against modification and fabrication by intruders is done through protocols for mutual authentication and message integrity.

Authentication

Before going into the details of various authentication protocols, it is worth-while noting that **authentication and message integrity cannot do without each other**. Consider, for example, a distributed system that supports authentication of two communicating parties, but does not provide mechanisms to ensure message integrity. In such a system, Bob may know for sure that Alice is the sender of a message m. However, if Bob cannot be given guarantees that m has not been modified during transmission, what use is it to him to know that Alice sent (the original version of) m?

Likewise, suppose that **only message integrity is supported, but no mechanisms exist for authentication**. When Bob receives a message stating that he has just won $1,000,000 in the lottery, how happy can he be if he cannot verify that the message was sent by the organizers of that lottery?

Consequently, **authentication and message integrity should go together**. In many protocols, the combination works roughly as follows. Again, assume that Alice and Bob want to communicate, and that Alice takes the initiative in setting up a channel. Alice starts by sending a message to Bob, or otherwise to a trusted third party who will help set up the channel. Once the channel has been set up, Alice knows for sure that she is talking to Bob, and Bob knows for sure he is talking to Alice, they can exchange messages.

To subsequently ensure integrity of the data messages that are exchanged after authentication has taken place, it is common practice to use secret-key cryptography by means of session keys. **A session key** is a shared (secret) key that is used to encrypt messages for integrity and possibly also confidentiality. Such a key is generally used only for as long as the channel exists. When the channel is closed, its associated session key is destroyed.

Authentication Methods

Different authentication methods are used based on different authentication algorithms. These authentication methods can be combined or used separately, depending on the level of functionality and security needed. Among such methods are:

- Password authentication
- Public-key authentication
- Anonymous authentication
- Remote authentication
- Certificate-based authentication

Message Integrity and Confidentiality

Beside authentication, a secure channel should also provide guarantees for message integrity and confidentiality. Message integrity means that messages are protected against surreptitious modification. Message Integrity is ensured by using digital signature process.

Confidentiality ensures that messages cannot be intercepted and read by eavesdroppers. Confidentiality is easily established by simply encrypting a message before sending it. Encryption can take place either through a secret key shared with the receiver or alternatively by using the receiver's public key.

9.3 Access Control

Once a client and a server have set up a secure channel, the client can issue requests that are to be carried out by the server. Requests involve carrying out operations on resources that are controlled by the server. A general situation is that of an object server that has a number of objects under its control. A request from a client generally involves invoking a method of a specific object. Such a request can be carried out only if the client has sufficient access rights for that invocation. Formally, verifying access rights is referred to as **access control**, whereas authorization is about granting access rights. The two terms are strongly related to each other and are often used in an interchangeable way.

General issues in access control

In order to understand the various issues involved in access control, the simple model shown in Figure 9.19 is generally adopted. It consists of subjects that issue a request to access an object. An object is very much like the objects we have been discussing so far. It can be thought of as encapsulating its own state and implementing the operations on that state. The operations of an object that subjects can request to be carried out are made available through interfaces. Subjects can best be thought of as being processes acting on behalf of users, but can also be objects that need the services of other objects in order to carry out their work.

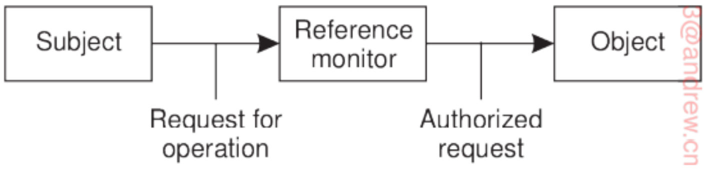


Figure 9.19: General model of controlling access to objects.

Controlling the access to an object is all about protecting the object against invocations by subjects that are not allowed to have specific (or even any) of the methods carried out. Also, protection may include object management issues, such as creating, renaming, or deleting objects. Protection is often enforced by a program called a reference monitor. A reference monitor records which subject may do what, and decides whether a subject is allowed to have a specific operation carried out. This monitor is called (e.g., by the underlying trusted operating system) each time an object is invoked. Consequently, it is extremely important that the reference monitor is itself tamperproof: an attacker must not be able to fool around with it.

Access Control Matrix

A common approach to modeling the access rights of subjects with respect to objects is to construct an access control matrix. Each subject is represented by a row in this matrix; each object is represented by a column.

One widely applied approach is to have each object maintain a list of the access rights of subjects that want to access the object. In essence, this means that the matrix is distributed column-wise across all objects, and that empty entries are left out. This type of implementation leads to what is called an **Access Control List (ACL).**

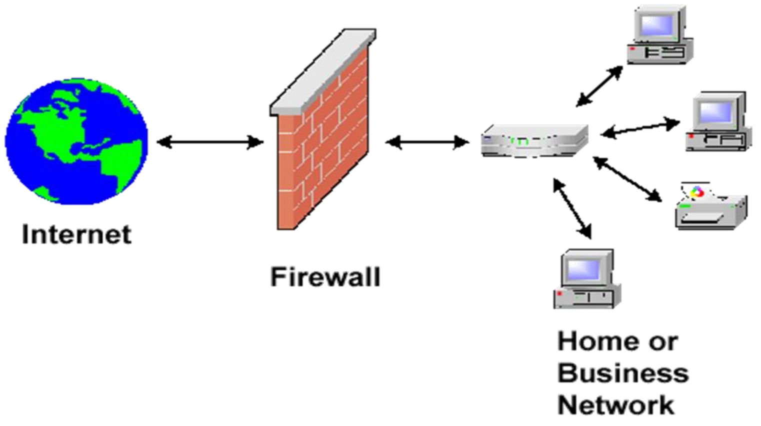
Table: Access Control Matrix

Users	Data Access File #1	Data Creation Application
A	Read/Write	Execute
B	Read	Execute
C	Non	None

Firewall

A **firewall** is a hardware or software designed to permit or deny network transmissions based upon a set of rules and is frequently used to **protect networks from unauthorized access** while permitting legitimate communications to pass.

A Firewall is hardware, software, or a combination of both which is used to prevent unauthorized programs or Internet users from accessing a private network and/or a single computer.



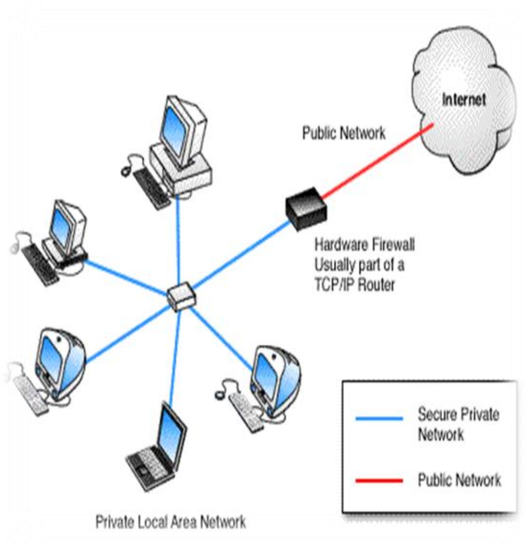
Hardware vs Software Firewall

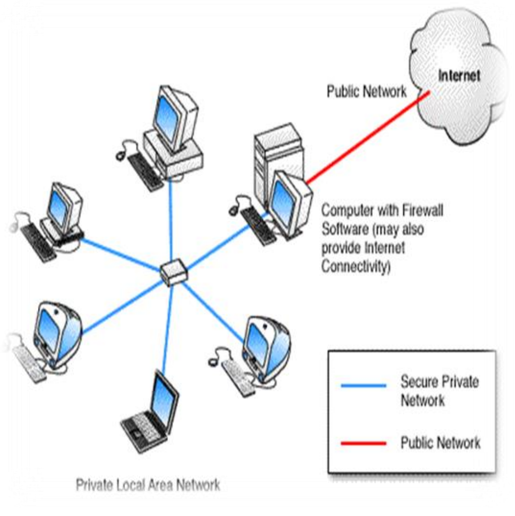
Hardware Firewalls

- Protect an entire network
- Implemented on the router level
- Usually more expensive, harder to configure

Software Firewalls

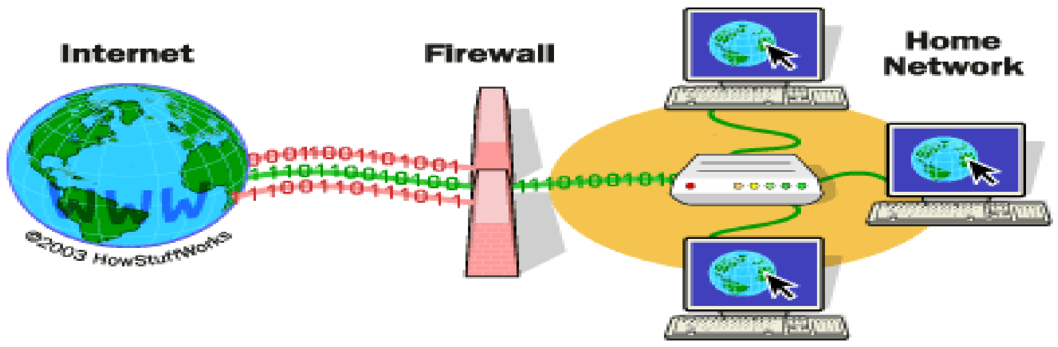
- Install in a single computer and protect all
- Usually less expensive, easier to configure


Internet
Public Network
Hardware Firewall Usually part of a TCP/IP Router
Secure Private Network
Public Network
Private Local Area Network


Internet
Public Network
Computer with Firewall Software (may also provide Internet Connectivity)
Secure Private Network
Public Network
Private Local Area Network

How does a software firewall work?

- Inspects each individual "packet" of data as it arrives at either side of the firewall.
- Determines whether it should be allowed to pass through or if it should be blocked.



Firewall Rules

Allow – traffic that flows automatically because it has been deemed
Block – traffic that is blocked because it has been deemed dangerous to your computer
Ask – asks the user whether or not the traffic is allowed to pass through

Characteristics of firewall

- Service control - Determines the types of Internet services that can be accessed, inbound or outbound.
- Direction control - Determines the direction in which particular service requests are allowed to flow.
- User control - Controls access to a service according to which user is attempting to access it
- Behavior control - Controls how particular services are used (e.g. filter e-mail)
- All traffic from inside to outside must pass through the firewall.
- Only authorized traffic will be allowed to pass.

What Can a Firewall Do?

- Focus for security decisions - Stop hackers from accessing your computer
- Can enforce security policy - Protects your personal information
- Limits your exposure - Blocks “pop up” ads and certain cookies
- Can log Internet activity efficiently - Determines which programs can access the Internet

What Can't a Firewall Do?

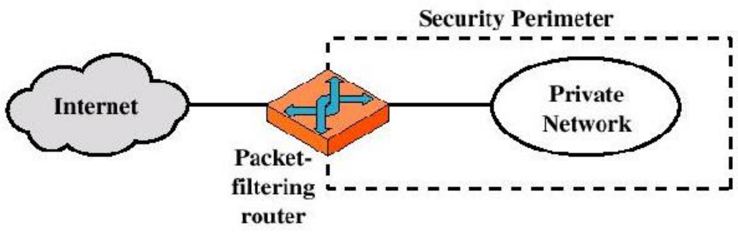
- Cannot protect against internal threats - For example, an angry employee deleting files Or, an employee cooperating with an outside attacker
- Cannot protect against attacks that bypass the firewall
- Can't protect against completely new threats
- Can't protect against viruses, different operating systems and applications inside the network - Need to scan all incoming data…impractical, perhaps impossible

Types of firewalls

Packet-filtering Router

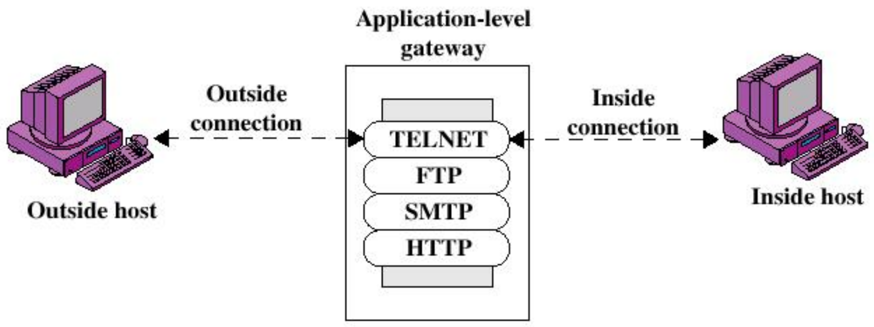
- Applies a set of rules to each incoming IP packet and then forwards or discards the packet
- Filter packets going in both directions

- The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header
- Two default policies (discard or forward)
- Advantages: Transparency to users, High speed
- Disadvantages: Difficulty of setting up packet filter rules, Lack of Authentication



Application-level Gateway

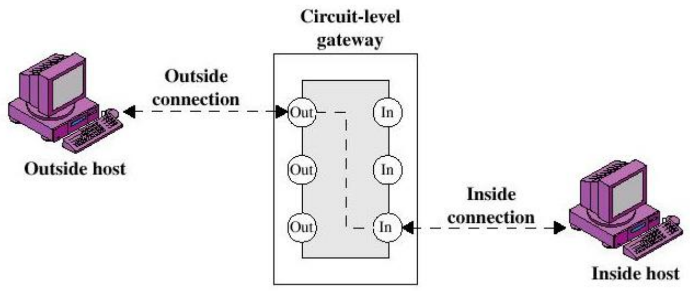
- Also called proxy server
- Acts as a relay of application-level traffic
- User contacts gateway through an application (e.g., telnet or FTP)
- User must authenticate and provide name of remote host
- Gateway connects to remote host and relays data back to the user
- If code for an application is not implemented, gateway will not support that application
- May be configured to support only certain features of an application
- Advantages: Higher security than packet filters, Only need to scrutinize a few allowable applications, Easy to log and audit all incoming traffic
- Disadvantages: Additional processing overhead on each connection (gateway as splice point)



Circuit-level Gateway

It provides session-level control over network traffic. Circuit-level gateways are host based and reside on individual clients and servers inside the network, rather than on a dedicated machine as they do with other types of firewalls. It examines incoming IP packets at the session level and act as relays by handing off incoming packets to other hosts. It monitors TCP handshaking between packets to determine whether a requested session is legitimate.

Circuit-level gateways are rarely used as a stand-alone firewall solution; instead, they are typically used in combination with application layer proxy services and packet filtering features in dedicated firewall applications.



Secure Mobile Code

An important issue in modern distributed systems is the ability to migrate code between hosts instead of just migrating passive data. However, mobile code introduces a number of serious security threats.

Two major issues are:

1. Securing against malicious agents that attempt to damage a mobile agent environment.
2. Securing a mobile agent from a malicious environment that attempts to interfere with the working of the mobile agent.

Denial of Service

Denial of Service (DoS) attacks are where the intruder attempts to crash a service (or the machine), overload network links, overload the CPU, or fill up the disk. Denial of service can also be caused by intentional degradation or blocking of computer or network resources. These denial-of-service attacks, commonly known as distributed denial-of-service (DDoS) attacks, are a new form of cyberattacks. They target computers connected to the Internet. They are not penetration attacks, and they do not change, alter, destroy, or modify system resources. However, they affect the system by diminishing the system's ability to function. Hence, they are capable of degrading the system's performance, eventually bringing a system down without destroying its resource.

9.4 Securing naming

A topic that has received increasingly more attention is that of secure naming. Simply put: when a client retrieves an object based on some name, how does it know that it got back the correct object? The whole issue is rather fundamental: when resolving a name in DNS, how does the client know it is returned the correct address? When looking up an object using some combination of a URL and database query, how does the receiver know it is returned what was requested? To be more precise, we have three issues to worry about [Smetters and Jacobson, 2009]:

1. **Validity**: is the object returned a complete, unaltered copy of what was stored at the server?
2. **Provenance**: can the server that returned the object be trusted as a genuine supplier? For example, it may be the case that a client is returned a cached version of the original object.
3. **Relevance**: is what was returned relevant considering what was asked?

A partial, and well-known solution to secure naming is to securely bind the name of an object to its content through hashing. Simply put: take hash (O) as the name of an object O. This is a form of a self-certifying name and at least allows the client to check for validity.

9.5 Security Management

In a security management:

- First, we need to consider the general management of cryptographic keys, and especially the means by which public keys are distributed. As it turns out, certificates play an important role here.
- Second, we discuss the problem of securely managing a group of servers by concentrating on the problem of adding a new group member that is trusted by the current members. Clearly, in the face of distributed and replicated services, it is important that security is not compromised by admitting a malicious process to a group.
- Third, we pay attention to authorization management by looking at capabilities and what are known as attribute certificates. An important issue in distributed systems with respect to authorization management is that one process can delegate some or all of its access rights to another process. Delegating rights in a secure way has its own subtleties.

Key Management

In the various cryptographic protocols, we assumed that various keys were readily available. For example, in the case of public-key cryptosystems, we assumed that a sender of a message had the public key of the receiver so that it could encrypt the message to ensure confidentiality. Likewise, in the case of authentication using a key distribution center (KDC), we assumed each party already shared a secret key with the KDC.

Cryptographic keys are a vital part of any security system. They do everything from data encryption and decryption and decryption to user authentication. Proper management of keys and their related components can ensure the safety of confidential information.

Key Management is the process of putting certain standards in place to ensure the security of cryptographic keys. It deals with the creation, exchange, storage, deletion, and refreshing of keys. There are two aspects for key management: **distribution of public keys** and use of **public-key encryption to distribute secret**.

Distribution of Public Key:

Public key can be distributed in 4 ways:

1. Public announcement – Public key is broadcast to everyone.
2. Publicly available directory – The public key is stored at a public directory.
3. Public-key Authority – It is similar to the directory but, improve security by tightening control over distribution of keys from directory.
4. Public-key certificate – Authority provides a certificate to allow key exchange without real-time access to the public authority each time.

Secure Group Management

Many security systems make use of special services such as Key Distribution Centers (KDCs) or Certification Authorities (CAs). These services demonstrate a difficult problem in distributed systems. In the first place, they must be trusted. To enhance the trust in security services, it is necessary to provide a high degree of protection against all kinds of security threats. For example, as soon as a CA has been compromised, it becomes impossible to verify the validity of a public key, making the entire security system worthless.

On the other hand, it is also necessary that many security services offer high availability. For example, in the case of a KDC, each time two processes want to set up a secure channel, at least one of them will need to contact the KDC for a shared secret key. If the KDC is not available, secure communication cannot be established unless an alternative technique for key establishment is available, such as the Diffie-Hellman key exchange.

The solution to high availability is replication. On the other hand, replication makes a server more vulnerable to security attacks. We already discussed how secure group communication can take place by sharing a secret among the group members. In effect, no single group member is capable of compromising certificates, making the group itself highly secure.

Authorization Management

Managing security in distributed systems is also concerned with managing access rights. So far, we have hardly touched upon the issue of how access rights are initially granted to users or groups of users, and how they are subsequently maintained in an unforgeable way. It is time to correct this omission.

In non-distributed systems, managing access rights is relatively easy. When a new user is added to the system, that user is given initial rights, for example, to create files and subdirectories in a specific directory, create processes, use CPU time, and so on. In other words, a complete account for a user is set up for one specific machine in which all rights have been specified in advance by the system administrators.

In a distributed system, matters are complicated by the fact that resources are spread across several machines. If the approach for non-distributed systems were to be followed, it would be necessary to create an account for each user on each machine. In essence, this is the approach followed in network operating systems. Matters can be simplified a bit by creating a single account on a central server. That server is consulted each time a user accesses certain resources or machines.

Practice Questions

1. Explain the security concerns in distributed system.
2. What is a security threat? Explain different types of security threats.
3. Discuss different design issues in distributed security.
4. Explain the encryption and decryption in cryptosystem.
5. Differentiate between symmetric-key cryptography and asymmetric-key cryptography.
6. What is secure channel? Explain the role of authentication, message integrity and confidentiality in secure channel of distributed system.
7. Define firewall. Write the advantages and disadvantages of firewall.
8. Explain the types of firewall.
9. Discuss about secure mobile code.
10. Explain about security management along with key management, secure group management and authorization management.
11. Write short notes on:
    a) Access control
    b) Secure naming