CMR
INSTITUTE OF
TECHNOLOGY

USN □□□□□□□□□□

Internal Assesment Test – III

Sub:	NETWORK SECURITY		Sec	A, C & D				Code:	10EC832
Date:	21-05-18	Duration:	90 mins	Max Marks:	50	Sem:	VIII	Branch:	ECE

Solution

Marks

1 What are the different firewall configurations? Explain the same. 10

Firewall Configurations

In addition to the use of a simple configuration consisting of a single system, such as a single packet-filtering router or a single gateway (Figure 1), more complex configurations are possible and indeed more common. Figure 1 illustrates three common firewall configurations. We examine each of these in turn.

In the screened host firewall, single-homed bastion configuration (Figure 1a), the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.
2. For traffic from the internal network, only IP packets from the bastion host are allowed out.

The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application-level gateway alone, for two reasons. First, this configuration implements both packet-level and application-level filtering, allowing for considerable flexibility in defining security policy. Second, an intruder must generally penetrate two separate systems before the security of the internal network is compromised.

This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet.

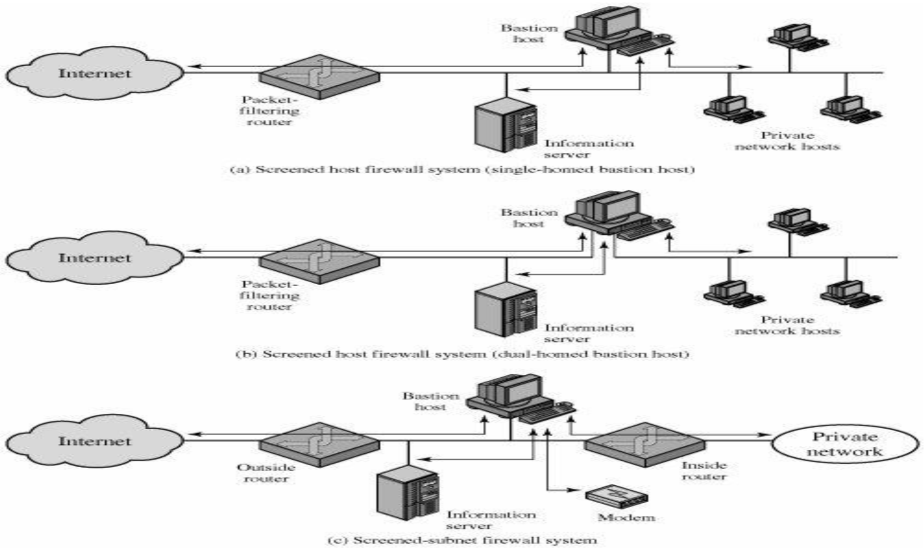
In the single-homed configuration just described, if the packet-filtering router is completely compromised, traffic could flow directly through the router between the Internet and other hosts on the private network. The screened host firewall, dual-homed bastion configuration physically prevents such a security breach (Figure 1b). The advantages of dual layers of security that were present in the previous configuration are present here as well. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy.

The screened subnet firewall configuration of Figure 1c is the most secure of those we have considered. In this configuration, two packet-filtering routers are used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked.

This configuration offers several advantages:

- There are now three levels of defense to thwart intruders.
- The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet.

Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet.



(a) Screened host firewall system (single-homed bastion host)

(b) Screened host firewall system (dual-homed bastion host)

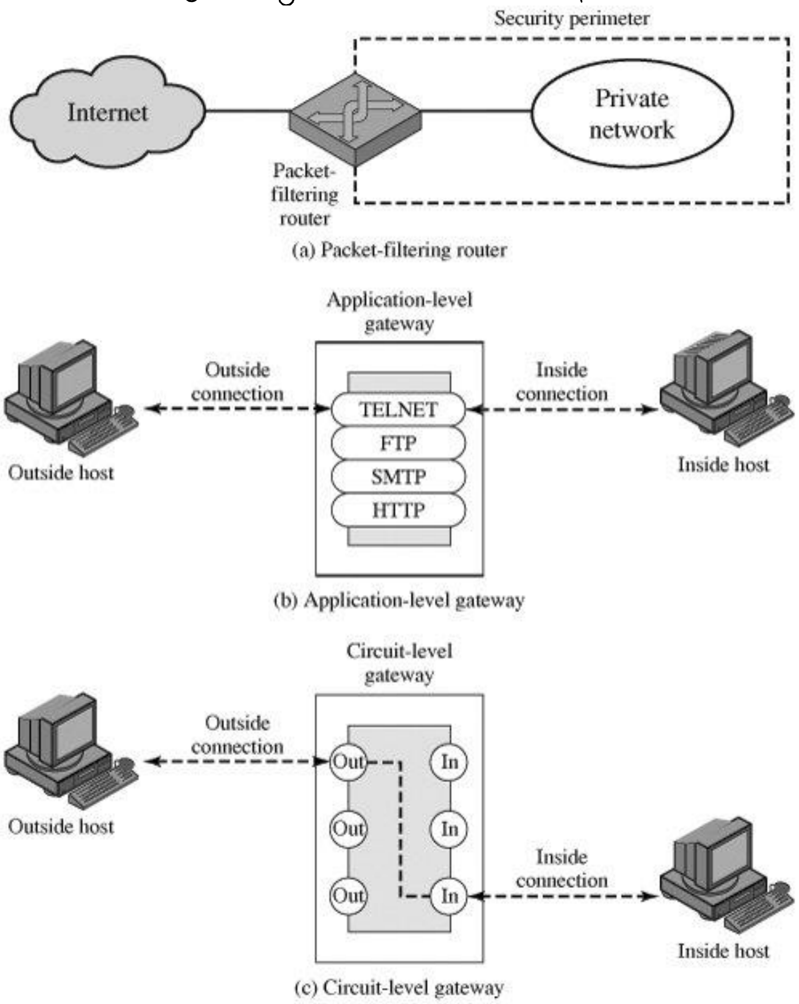
(c) Screened-subnet firewall system

Figure 1. Firewall Configurations

2 What are the different types of firewall? Explain packet filtering firewall. 10

Types of Firewalls

Figure 2 illustrates the three common types of firewalls: packet filters, application-level gateways, and circuit-level gateways. We examine each of these in turn.



(a) Packet-filtering router

(b) Application-level gateway

(c) Circuit-level gateway

Figure 2 Firewall Types

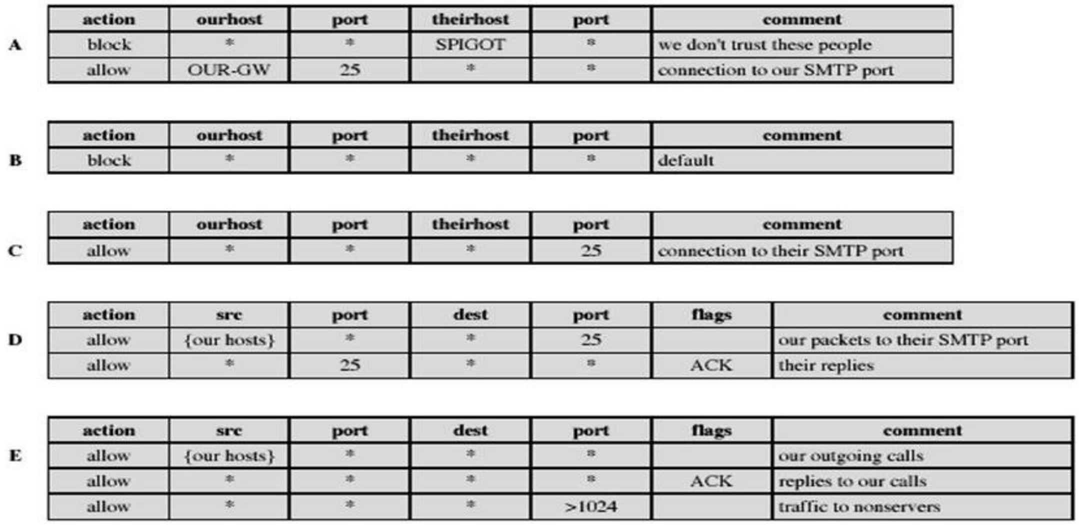
Packet-Filtering Router

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:

- *Source IP address:* The IP address of the system that originated the IP packet (e.g.,192.172.1)
- *Destination IP address:* The IP address of the system the IP packet is trying to reach (e.g.,192.162.2)
- *Source and destination transport-level address:* The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET
- *IP protocol field:* Defines the transport protocol
- *Interface:* For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. Two default policies are possible:

    - Default = discard: That which is not expressly permitted is prohibited.
    - Default = forward: That which is not expressly prohibited is permitted.

Table 2 Packet filtering Examples



A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

The default discard policy is more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are more likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security; the security administrator must, in essence, react to each new security threat as it becomes known.

Table 2, from [BELL94b], gives some examples of packet-filtering rule sets. In each set, the rules are applied top to bottom. The "*" in a field is a wildcard designator that matches everything. We assume that the default = discard policy is in force.

A. Inbound mail is allowed (port 25 is for SMTP incoming), but only to a gateway host. However, packets from a particular external host, SPIGOT, are blocked because that host has a history of sending massive files in e-mail messages.
B. This is an explicit statement of the default policy. All rule sets include this rule implicitly as the last rule.
C. This rule set is intended to specify that any inside host can send mail to the outside. A TCP packet with a destination port of 25 is routed to the SMTP server on the destination machine. The problem with this rule is that the use of port 25 for SMTP receipt is only a default; an outside machine could be configured to

have some other application linked to port 25. As this rule is written, an attacker could gain access to internal machines by sending packets with a TCP source port number of 25.

D. This rule set achieves the intended result that was not achieved in C. The rules take advantage of a feature of TCP connections. Once a connection is set up, the ACK flag of a TCP segment is set to acknowledge segments sent from the other side. Thus, this rule set states that it allows IP packets where the source IP address is one of a list of designated internal hosts and the destination TCP port number is 25. It also allows incoming packets with a source port number of 25 that include the ACK flag in the TCP segment. Note that we explicitly designate source and destination systems to define these rules explicitly.

E. This rule set is one approach to handling FTP connections. With FTP, two TCP connections are used: a control connection to set up the file transfer and a data connection for the actual file transfer. The data connection uses a different port number that is dynamically assigned for the transfer. Most servers, and hence most attack targets, live on low-numbered ports; most outgoing calls tend to use a higher-numbered port, typically above 1023. Thus, this rule set allows
    - Packets that originate internally
    - Reply packets to a connection initiated by an internal machine
    - Packets destined for a high-numbered port on an internal machine

This scheme requires that the systems be configured so that only the appropriate port numbers are in use.

Rule set E points out the difficulty in dealing with applications at the packet-filtering level. Another way to deal with FTP and similar applications is an application-level gateway, described later in this section.

One advantage of a packet-filtering router is its simplicity. Also, packet filters typically are transparent to users and are very fast. [WACK02] lists the following weaknesses of packet filter firewalls:

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions. For example, a packet filter firewall cannot block specific application commands; if a packet filter firewall allows a given application, all functions available within that application will be permitted.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited. Packet filter logs normally contain the same information used to make access control decisions (source address, destination address, and traffic type).
- Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall.
- They are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as *network layer address spoofing*. Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered. Spoofing attacks are generally employed by intruders to bypass the security controls implemented in a firewall platform.
- Finally, due to the small number of variables used in access control decisions, packet filter firewalls are susceptible to security breaches caused by improper configurations. In other words, it is easy to accidentally configure a packet filter

firewall to allow traffic types, sources, and destinations that should be denied based on an organization's information security policy.

Some of the attacks that can be made on packet-filtering routers and the appropriate countermeasures are the following:

- *IP address spoofing:* The intruder transmits packets from the outside with a source IP address field containing an address of an internal host. The attacker hopes that the use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from specific trusted internal hosts are accepted. The countermeasure is to discard packets with an inside source address if the packet arrives on an external interface.
- *Source routing attacks:* The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. The countermeasure is to discard all packets that use this option.
- *Tiny fragment attacks:* The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. This attack is designed to circumvent filtering rules that depend on TCP header information. Typically, a packet filter will make a filtering decision on the first fragment of a packet. All subsequent fragments of that packet are filtered out solely on the basis that they are part of the packet whose first fragment was rejected. The attacker hopes that the filtering router examines only the first fragment and that the remaining fragments are passed through. A tiny fragment attack can be defeated by enforcing a rule that the first fragment of a packet must contain a predefined minimum amount of the transport header. If the first fragment is rejected, the filter can remember the packet and discard all subsequent fragments.

3 Write a note on data access control and explain the concept of trusted system. 10

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology. This section provides a brief overview of this topic. We begin by looking at some basic concepts of data access control.

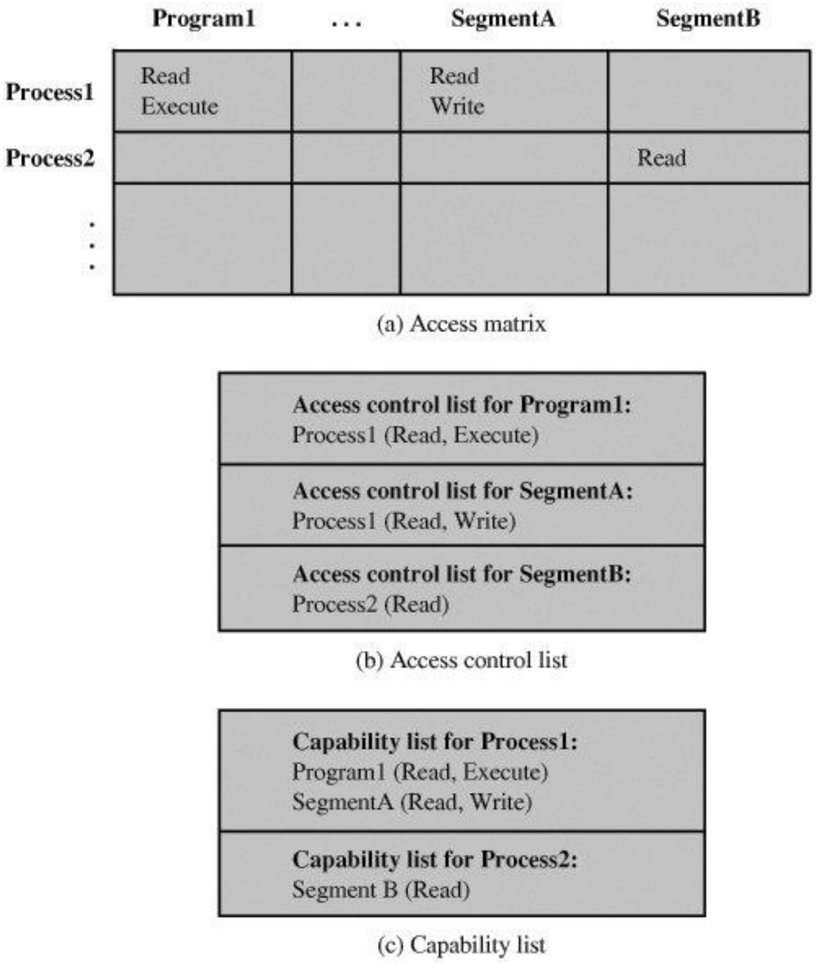
Data Access Control

Following successful logon, the user has been granted access to one or a set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible operations and file accesses. The operating system can then enforce rules based on the user profile. The database management system, however, must control access to specific records or even portions of records. For example, it may be permissible for anyone in administration to obtain a list of company personnel, but only selected individuals may have access to salary information. The issue is more than just one of level of detail. Whereas the operating system may grant a user permission to access a file or use an application, following which there are no further security checks, the database management system must make a decision on each individual access attempt. That decision will depend not only on the user's identity but also on the specific parts of the data being accessed and even on the information already divulged to the user.

A general model of access control as exercised by a file or database management system is that of an access matrix (Figure 3aa). The basic elements of the model are as follows:

- *Subject:* An entity capable of accessing objects. Generally, the concept of subject equates with that of process. Any user or application actually gains access to an object by means of a process that represents that user or application.
- *Object:* Anything to which access is controlled. Examples include files, portions of files, programs, and segments of memory.
- *Access right:* The way in which an object is accessed by a subject. Examples are read, write, and execute.

One axis of the matrix consists of identified subjects that may attempt data access. Typically, this list will consist of individual users or user groups, although access could be controlled for terminals, hosts, or applications instead of or in addition to users. The other axis lists the objects that may be accessed. At the greatest level of detail, objects may be individual data fields. More aggregate groupings, such as records, files, or even the entire database, may also be objects in the matrix. Each entry in the matrix indicates the access rights of that subject for that object.



	Program1	. . .	SegmentA	SegmentB
Process1	Read Execute		Read Write	
Process2				Read
⋮				

(a) Access matrix

Access control list for Program1:
Process1 (Read, Execute)

Access control list for SegmentA:
Process1 (Read, Write)

Access control list for SegmentB:
Process2 (Read)

(b) Access control list

Capability list for Process1:
Program1 (Read, Execute)
SegmentA (Read, Write)

Capability list for Process2:
Segment B (Read)

(c) Capability list

Figure 3a. Access Control Structure

In practice, an access matrix is usually sparse and is implemented by decomposition in one of two ways. The matrix may be decomposed by columns, yielding access control lists (Figure 3ab). Thus, for each object, an access control list lists users and their permitted access rights. The access control list may contain a default, or public, entry. This allows users that are not explicitly listed as having special rights to have a default set of rights. Elements of the list may include individual users as well as groups of users. Decomposition by rows yields capability tickets (Figure 3ac). A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to loan or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than access control lists. In particular, the ticket must be unforgeable. One way to accomplish this is to have the operating system hold all tickets on behalf of users. These tickets would have to be held in a region of memory inaccessible to users.

The Concept of Trusted Systems

Much of what we have discussed so far has been concerned with protecting a given message or item from passive or active attacks by a given user. A somewhat different but widely applicable requirement is to protect data or resources on the basis of levels of security. This is commonly found in the military, where information is categorized as unclassified (U), confidential (C), secret (S), top secret (TS), or beyond. This concept is equally applicable in other areas, where information can be organized into gross categories and users can be granted clearances to access certain categories of data. For example, the highest level of security might be for strategic corporate planning documents and data, accessible by only corporate officers and their staff; next might come sensitive financial and personnel data, accessible only by administration personnel, corporate officers, and so on.

When multiple categories or levels of data are defined, the requirement is referred to as multilevel security. The general statement of the requirement for multilevel security is that a subject at a high level may not convey information to a subject at a lower or non-comparable level unless that flow accurately reflects the will of an authorized user. For implementation purposes, this requirement is in two parts and is simply stated. A multilevel secure system must enforce the following:

- No read up: A subject can only read an object of less or equal security level. This is referred to in the literature as the Simple Security Property.
- No write down: A subject can only write into an object of greater or equal security level. This is referred to in the literature as the *-Property[1] (pronounced *star property*). [1]The "*" does not stand for anything. No one could think of an appropriate name for the property during the writing of the first report on the model. The asterisk was a dummy character entered in the draft so that a text editor could rapidly find and replace all instances of its use once the property was named. No name was ever devised, and so the report was published with the "*" intact.

These two rules, if properly enforced, provide multilevel security. For a data processing system, the approach that has been taken, and has been the object of much research and development, is based on the *reference monitor* concept. This approach is depicted in Figure 3b. The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of security parameters of the subject and object. The reference monitor has access to a file, known as the *security kernel database* that lists the access privileges (security clearance) of each subject and the protection attributes (classification level) of each object. The reference monitor enforces the security rules (no read up, no write down) and has the following properties:

- Complete mediation: The security rules are enforced on every access, not just, for example, when a file is opened.
- Isolation: The reference monitor and database are protected from unauthorized modification.
- Verifiability: The reference monitor's correctness must be provable. That is, it must be possible to demonstrate mathematically that the reference monitor enforces the security rules and provides complete mediation and isolation.

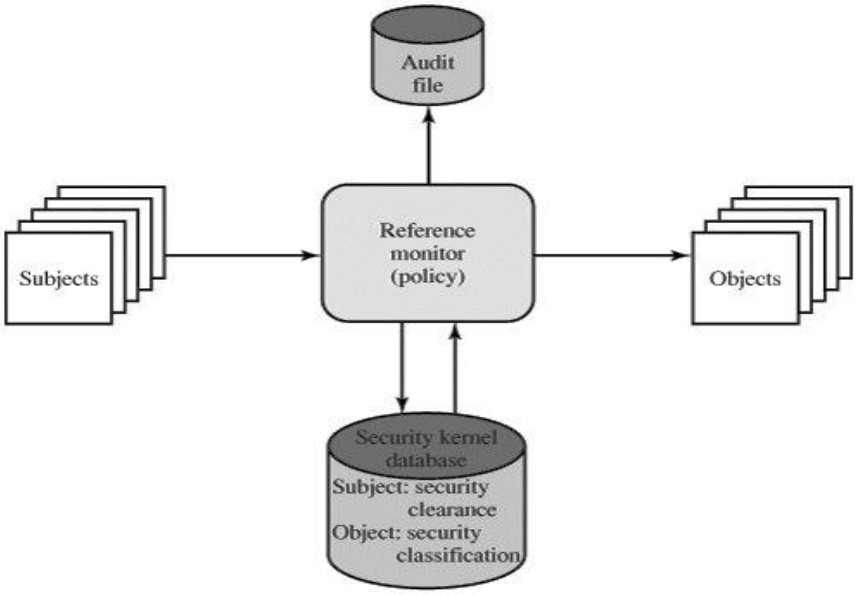

Audit file
Subjects
Reference monitor (policy)
Objects
Security kernel database
Subject: security clearance
Object: security classification

Figure 3b. Reference Monitor Concept

These are stiff requirements. The requirement for complete mediation means that every access to data within main memory and on disk and tape must be mediated. Pure software implementations impose too high a performance penalty to be practical; the solution must be at least partly in hardware. The requirement for isolation means that it must not be possible for an attacker, no matter how clever, to change the logic of the reference monitor or the contents of the security kernel database. Finally, the requirement for mathematical proof is formidable for something as complex as a general-purpose computer. A system that can provide such verification is referred to as a trusted system. A final element illustrated in Figure 3b is an audit file. Important security events, such as detected security violations and authorized changes to the security kernel database, are stored in the audit file.

In an effort to meet its own needs and as a service to the public, the U.S. Department of Defense in 1981 established the Computer Security Center within the National Security Agency (NSA) with the goal of encouraging the widespread availability of trusted computer systems. This goal is realized through the center's Commercial Product Evaluation Program. In essence, the center attempts to evaluate commercially available products as meeting the security requirements just outlined. The center classifies evaluated products according to the range of security features that they provide. These evaluations are needed for Department of Defense procurements but are published and freely available. Hence, they can serve as guidance to commercial customers for the purchase of commercially available, off-the-shelf equipment.

4 Explain with necessary diagrams the trojan horse defense. 10

Trojan Horse Defense

One way to secure against Trojan horse attacks is the use of a secure, trusted operating system. Figure 4 illustrates an example. In this case, a Trojan horse is used to get around the standard security mechanism used by most file management and operating systems: the access control list. In this example, a user named Bob interacts through a program with a data file containing the critically sensitive character string "CPE170KS." User Bob has created the file with read/write permission provided only to programs executing on his own behalf: that is, only processes that are owned by Bob may access the file.

The Trojan horse attack begins when a hostile user, named Alice, gains legitimate access to the system and installs both a Trojan horse program and a private file to be used in the attack as a "back pocket." Alice gives read/write permission to herself for this file and gives Bob write-only permission (Figure 4a). Alice now induces Bob to invoke the Trojan horse program, perhaps by advertising it as a useful utility. When the program detects that it is being executed by Bob, it reads the sensitive character string from Bob's file and copies it into Alice's back-pocket file (Figure 4b). Both the read and write operations satisfy the constraints imposed by access control lists. Alice then has only to access Bob's file at a later time to learn the value of the string.

Now consider the use of a secure operating system in this scenario (Figure 4c). Security levels are assigned to subjects at logon on the basis of criteria such as the terminal from which the computer is being accessed and the user involved, as identified by password/ID. In this example, there are two security levels, sensitive and public, ordered so that sensitive is higher than public. Processes owned by Bob and Bob's data file are assigned the security level sensitive. Alice's file and processes are restricted to public. If Bob invokes the Trojan horse program (Figure 4d), that program acquires Bob's security level. It is therefore able, under the simple security property, to observe the sensitive character string. When the program attempts to store the string in a public file (the back-pocket file), however, the is violated and the attempt is disallowed by the reference monitor. Thus, the attempt to write into the back-pocket file is denied even though the access control list permits it: The security policy takes precedence over the access control list mechanism.

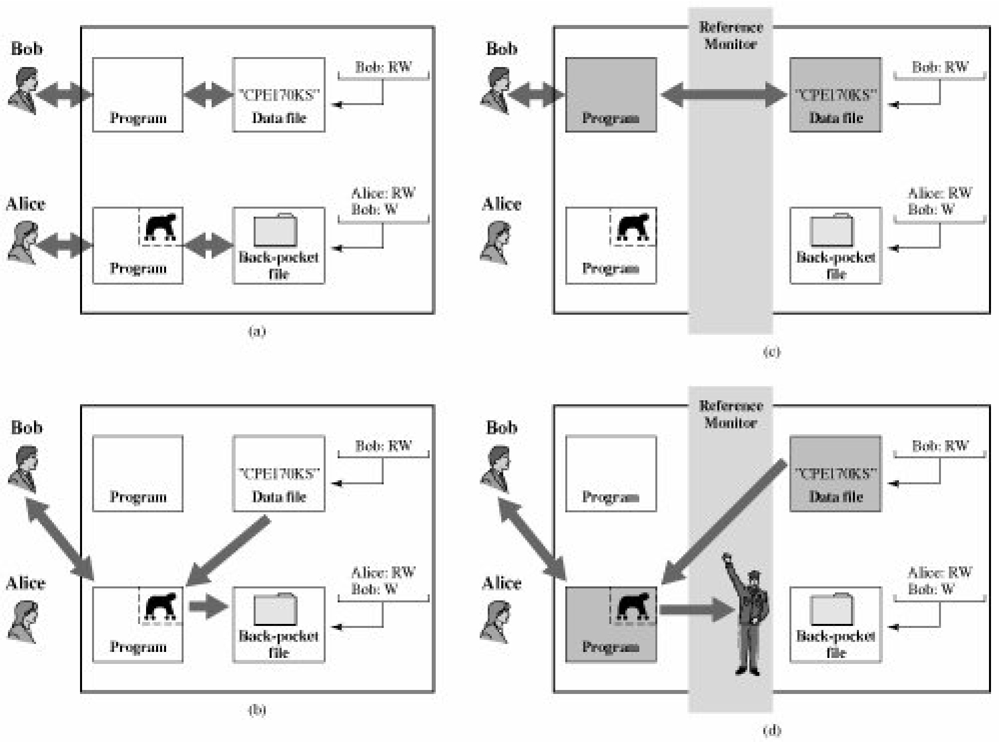


Figure 4. Trojan Horse and Secure Operating System

5 Describe the sequence of events that are required for a transaction using SET. 10

The sequence of events that are required for a transaction. We will then look at some of the cryptographic details.

1. ***The customer opens an account.*** The customer obtains a credit card account, such as MasterCard or Visa, with a bank that supports electronic payment and SET.

2. ***The customer receives a certificate.*** After suitable verification of identity, the customer receives an X.509v3 digital certificate, which is signed by the bank. The certificate verifies the customer's RSA public key and its expiration date. It also establishes a relationship, guaranteed by the bank, between the customer's key pair and his or her credit card.
3. ***Merchants have their own certificates.*** A merchant who accepts a certain brand of card must be in possession of two certificates for two public keys owned by the merchant: one for signing messages, and one for key exchange. The merchant also needs a copy of the payment gateway's public-key certificate.
4. ***The customer places an order.*** This is a process that may involve the customer first browsing through the merchant's Web site to select items and determine the price. The customer then sends a list of the items to be purchased to the merchant, who returns an order form containing the list of items, their price, a total price, and an order number.
5. ***The merchant is verified.*** In addition to the order form, the merchant sends a copy of its certificate, so that the customer can verify that he or she is dealing with a valid store.
6. ***The order and payment are sent.*** The customer sends both order and payment information to the merchant, along with the customer's certificate. The order confirms the purchase of the items in the order form. The payment contains credit card details. The payment information is encrypted in such a way that it cannot be read by the merchant. The customer's certificate enables the merchant to verify the customer.
7. ***The merchant requests payment authorization.*** The merchant sends the payment information to the payment gateway, requesting authorization that the customer's available credit is sufficient for this purchase.
8. ***The merchant confirms the order.*** The merchant sends confirmation of the order to the customer.
9. ***The merchant provides the goods or service.*** The merchant ships the goods or provides the service to the customer.
10. ***The merchant requests payment.*** This request is sent to the payment gateway, which handles all of the payment processing.

6 Explain in detail the purchase request and payment authorization transaction. 10

Purchase Request

Before the Purchase Request exchange begins, the cardholder has completed browsing, selecting, and ordering. The end of this preliminary phase occurs when the merchant sends a completed order form to the customer. All of the preceding occurs without the use of SET.

The purchase request exchange consists of four messages: Initiate Request, Initiate Response, Purchase Request, and Purchase Response.

In order to send SET messages to the merchant, the cardholder must have a copy of the certificates of the merchant and the payment gateway. The customer requests the certificates in the Initiate Request message, sent to the merchant. This message includes the brand of the credit card that the customer is using. The message also includes an ID assigned to this request/response pair by the customer and a nonce used to ensure timeliness.

The merchant generates a response and signs it with its private signature key. The response includes the nonce from the customer, another nonce for the customer to return in the next message, and a transaction ID for this purchase transaction. In addition to the signed response, the Initiate Response message includes the merchant's signature certificate and the payment gateway's key exchange certificate. The cardholder verifies the merchant and gateway certificates by means of their respective CA signatures and then creates the OI and PI.

The transaction ID assigned by the merchant is placed in both the OI and PI. The OI does not contain explicit order data such as the number and price of items. Rather, it contains an order reference generated in the exchange between merchant and customer during the shopping phase before the first SET message. Next, the cardholder prepares the Purchase Request message (Figure 6). For this purpose, the cardholder generates a one-time symmetric encryption key, *Ks*. The message includes the following:

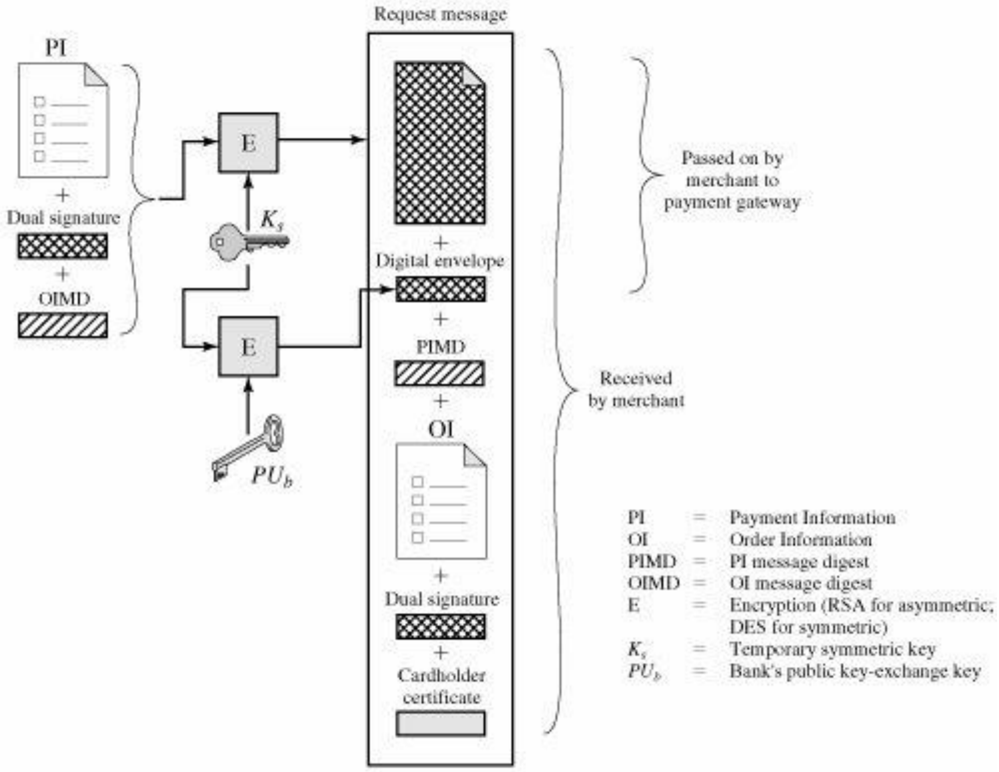


Figure 6a Cardholder sends purchase request

1. ***Purchase-related information.*** This information will be forwarded to the payment gateway by the merchant and consists of
   - The PI
   - The dual signature, calculated over the PI and OI, signed with the customer's private signature key
   - The OI message digest (OIMD)

The OIMD is needed for the payment gateway to verify the dual signature, as explained previously. All of these items are encrypted with *Ks*. The final item is

- The digital envelope. This is formed by encrypting *Ks* with the payment gateway's public key-exchange key. It is called a digital envelope because this envelope must be opened (decrypted) before the other items listed previously can be read.

The value of Ks is not made available to the merchant. Therefore, the merchant cannot read any of this payment-related information.

2. ***Order-related information.*** This information is needed by the merchant and consists of
   - The OI
   - The dual signature, calculated over the PI and OI, signed with the customer's private signature key
   - The PI message digest (PIMD)

The PIMD is needed for the merchant to verify the dual signature. Note that the OI is sent in the clear.

3. ***Cardholder certificate.*** This contains the cardholder's public signature key. It is needed by the merchant and by the payment gateway.

When the merchant receives the Purchase Request message, it performs the following actions (Figure 6b):

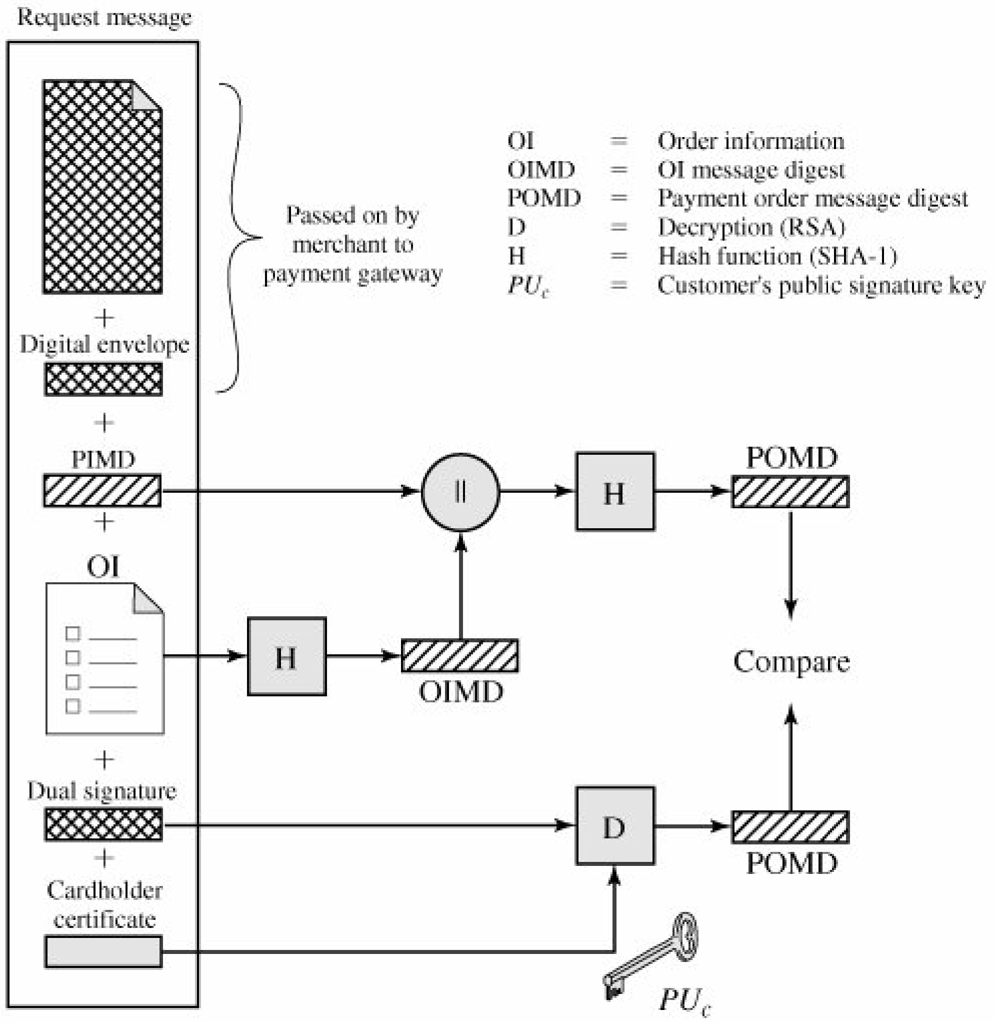


Figure 6b Merchant verifies customer purchase request

1. Verifies the cardholder certificates by means of its CA signatures.
2. Verifies the dual signature using the customer's public signature key. This ensures that the order has not been tampered with in transit and that it was signed using the cardholder's private signature key.

3. Processes the order and forwards the payment information to the payment gateway for authorization (described later).
4. Sends a purchase response to the cardholder.

The Purchase Response message includes a response block that acknowledges the order and references the corresponding transaction number. This block is signed by the merchant using its private signature key. The block and its signature are sent to the customer, along with the merchant's signature certificate.

When the cardholder software receives the purchase response message, it verifies the merchant's certificate and then verifies the signature on the response block. Finally, it takes some action based on the response, such as displaying a message to the user or updating a database with the status of the order.

Payment Authorization

During the processing of an order from a cardholder, the merchant authorizes the transaction with the payment gateway. The payment authorization ensures that the transaction was approved by the issuer. This authorization guarantees that the merchant will receive payment; the merchant can therefore provide the services or goods to the customer. The payment authorization exchange consists of two messages: Authorization Request and Authorization response.

The merchant sends an Authorization Request message to the payment gateway consisting of the following:

1. **Purchase-related information.** This information was obtained from the customer and consists of
   - The PI
   - The dual signature, calculated over the PI and OI, signed with the customer's private signature key
   - The OI message digest (OIMD)
   - The digital envelope

2. **Authorization-related information.** This information is generated by the merchant and consists of
   - An authorization block that includes the transaction ID, signed with the merchant's private signature key and encrypted with a one-time symmetric key generated by the merchant.
   - A digital envelope. This is formed by encrypting the one-time key with the payment gateway's public key-exchange key.
3. **Certificates.** The merchant includes the cardholder's signature key certificate (used to verify the dual signature), the merchant's signature key certificate (used to verify the merchant's signature), and the merchant's key-exchange certificate (needed in the payment gateway's response).

The payment gateway performs the following tasks:

1. Verifies all certificates
2. Decrypts the digital envelope of the authorization block to obtain the symmetric key and then decrypts the authorization block
3. Verifies the merchant's signature on the authorization block
4. Decrypts the digital envelope of the payment block to obtain the symmetric key and then decrypts the payment block
5. Verifies the dual signature on the payment block

6. Verifies that the transaction ID received from the merchant matches that in the PI received (indirectly) from the customer
7. Requests and receives an authorization from the issuer

Having obtained authorization from the issuer, the payment gateway returns an Authorization Response message to the merchant. It includes the following elements:

1. ***Authorization-related information.*** Includes an authorization block, signed with the gateway's private signature key and encrypted with a one-time symmetric key generated by the gateway. Also includes a digital envelope that contains the one-time key encrypted with the merchants public key-exchange key.
2. ***Capture token information.*** This information will be used to effect payment later. This block is of the same form as (1), namely, a signed, encrypted capture token together with a digital envelope. This token is not processed by the merchant. Rather, it must be returned, as is, with a payment request.
3. ***Certificate.*** The gateway's signature key certificate.

With the authorization from the gateway, the merchant can provide the goods or service to the customer.

7 List and explain the general approach to deal with replay attacks. 10

One approach to coping with replay attacks is to attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order.

The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with. Because of this overhead, sequence numbers are generally not used for authentication and key exchange. Instead, one of the following two general approaches is used:

- ***Timestamps:*** Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.
- ***Challenge/response:*** Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the Authentication Protocols correct nonce value.

It can be argued (e.g., [LAM92a]) that the timestamp approach should not be used for connection oriented applications because of the inherent difficulties with this technique. First, some sort of protocol is needed to maintain synchronization among the various processor clocks. This protocol must be both fault tolerant, to cope with network errors, and secure, to cope with hostile attacks. Second, the opportunity for a successful attack will arise if there is a temporary loss of synchronization resulting from a fault in the clock mechanism of one of the parties. Finally, because of the variable and unpredictable nature of network delays, distributed clocks cannot be expected to maintain precise synchronization. Therefore, any timestamp-based procedure must allow for a window of time sufficiently large to accommodate network delays yet sufficiently small to minimize the opportunity for attack.

On the other hand, the challenge-response approach is unsuitable for a connectionless type of application because it requires the overhead of a handshake before any connectionless transmission, effectively negating the chief characteristic of a connectionless transaction. For such applications, reliance on some sort of secure time server and a consistent attempt by each party to keep its clocks in synchronization may be the best approach.