Trusted Systems

Another approach to protection data and resources is based on levels of security. This is commonly found in 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 categories and users can be granted clearances to access certain categories of data. 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 noncomparable 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 write into an object of greater or equal security level. This is referred to as the ***-property** (pronounced star property)

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. 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 privilege (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:

Trusted Systems (Cont 1)

- Complete mediation: The security rules are enforced on every access, not just, for example, when a file is opened (requires high performance overhead)
- Isolation: The reference monitor and database are protected from unauthorized modification (requires impossibility for attacker to change database)
- 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 provided complete mediation and isolation If provided, system is referred to as a **trusted system**)

These are stiff requirements.
Important security events, such as detected security violations and authorized changes to the security database, are stored in the audit file.

Trojan Horse Defense

One way to secure against Trojan horse attacks is the use of a secure, trusted operating system. Figure 15.10 illustrates an example, when 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 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 and gives Bob write-only permission (Fig. 15.10a). 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 15.10b). Both the read

Trojan Horse Defense (Cont 1)

and write operations satisfy the constraints imposed by access control lists. Alice then has only to access her file at a later time to learn the value of the string.

Now consider the use of a secure operating system in this scenario (Fig. 15.10c). 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 (gray) and public (white), 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 files and processes are restricted to public. If Bob invokes the Trojan horse program (Fig. 15.10d), that program acquires Bob's level security. 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 *-property 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.

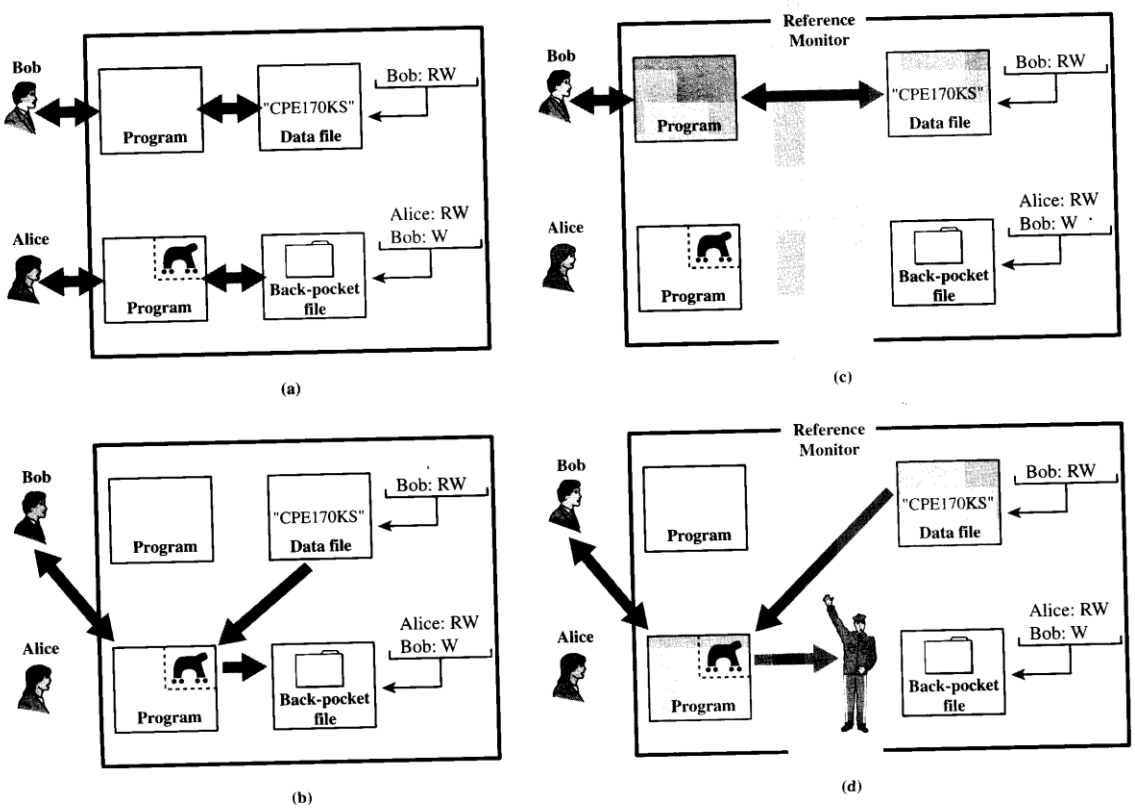

Figure 15.10 Trojan Horse and Secure Operating System

Types of Vulnerabilities

Category	Description	Examples
Installation and configuration	This type of vulnerability results from using a default installation and configuration that is known publicly and usually does not enforce any security measures. Also, improper configuration or installation may result in security risks	Incorrect application configuration that may result in application malfunction Failure to change default password Failure to change default permissions and privileges Use of default configuration in most cases do not provide security measures
Software	These are related to carelessness in implementing software	Lack of auditing controls Untested disaster recovery plan Lack of activity control Lack of protection against malicious code Bad authentication process implementation Software bugs Input data is not validated Exceptional conditions are not handled
User mistakes	These are due to not enough education of users	Lack of applying patches as they are released Social engineering (pretending to be a representative of a legitimate organization to trick an individual into providing sensitive information) A user's lack of technical information that leads to user susceptibility to various hacker intrusions and fraud schemes

Authentication, Authorization, and Encryption

Authentication refers to the process by which the identity of a participant is established
Authorization refers to the process that determines the mode in which a particular client is allowed to access a specific resource controlled by a server.
Encryption is used to protect information from being accessed by unauthorized users

Encryption

Plaintext is the original text
Ciphertext C is the result of encryption of a plaintext P using **key** K_1:

$$C = E_{K_1}(P)$$

The complete encryption system, including encryption and decryption, can be represented by:

$$P = D_{K_2}(E_{K_1}(P))$$

Which states that if the plaintext is first encrypted with K_1 and then decrypted with K_2, the result will be the original plaintext

Symmetric cryptography

With symmetric cryptography, $K_1 = K_2$. Key is often referred to as a session key. Generally, each session gets a new **session** key.
With a **block** cipher, the plaintext is divided into fixed-size blocks. A substitution cipher is one type of a block cipher, for example, **monoalphabetic** substitution cipher has a block size of one character.
Polyalphabetic cipher user several alphabets for substitution
A **polygram** substitution uses block size of several symbols
A **transposition** cipher is also a block cipher but the order of the characters in each plaintext is altered to produce the ciphertext block – the reordering is the same on each plaintext block and is described by the key.
ANSI's **Data Encryption Standard (DES)** is a symmetric encryption technique that uses a sequence of stages to encrypt a block of plaintext. It uses substitutions and transpositions (permutations).
A **bit stream** cipher produces ciphertext by bit-by-bit exclusive OR on the plaintext and a pseudorandom sequence of bits produced by a **pseudo random number generator**:

$$C_i = P_i \oplus R_i$$

Asymmetric cryptography uses two different keys: $K_1 \neq K_2$. One key is known as a public key, and the other as a private key. A **private** key is kept in secret, while a **public** key is made available for public. Asymmetric cryptography is also called **public-key** cryptography.

RSA algorithm

RSA (Rivest-Shamir-Adelman, 1978) algorithm is an asymmetric encryption algorithm. To design an encryption/decryption key pair, two large prime numbers, p and q, $p \neq q$, are selected, and an integer, d, is chosen that is relatively prime to (p-1)(q-1) (d and (p-1)(q-1) have no common factors other than 1). Finally, an integer e is computed such that

$$e \cdot d \equiv 1(\text{mod}(p-1) \cdot (q-1))$$

One key is (e,N), and the other is (d,N), where N=p*q, and is referred to as the modulus.
For example, we might select p=7, and q=13. Then N=91, and (p-1)(q-1)=72. We can choose d=5 (which is relatively prime to 72) and e=29, because e*d=145 and

$$145 \equiv 1(\text{mod } 72)$$

Then, one key is K_1=(29,91) and the other is K_2=(5,91). The message to be encrypted is broken into blocks such that each block, M, can be treated as an integer between 0 and (N-1). To encrypt M into the ciphertext block, B, we perform

$$B = M^{K_1} (\text{mod } N)$$

To decrypt B, we perform

$$M = B^{K_2} (\text{mod } N)$$

The protocol works correctly because

$$M = (M^{K_1} (\text{mod } N))^{K_2} (\text{mod } N) = M^{K_1 K_2} (\text{mod } N)$$

More details about RSA algorithm can be found in the textbook by William Stallings, Cryptography and Network Security.
Returning to the example, assume M=2.
Then, to encrypt M, we compute

$$2^{29} (\text{mod } 91) = 32$$

Thus, B=32. To decrypt B, we compute

$$32^5 (\text{mod } 91) = 2$$

which is the plaintext message M.
Obtaining of p and q is extremely difficult, hence, only knowing a secret key K_2, receiver can correctly decrypt a message.