Security and Protection

SECURITY
involves guarding computer resources against unauthorized access, malicious destruction or alteration, and accidental introduction of inconsistency.

- a system is **secure** if its resources are used and accessed as intended under all circumstances.
- security violations are either intentional or accidental
- several forms of security violations: breach of confidentiality, breach of integrity, breach of availability, theft of service, denial of service
- breach of confidentiality: involves unauthorized **reading** of data e.g. credit-card info, identity info
- breach of integrity: involves unauthorized **modification** of data e.g. hacking into open-source application
- breach of availability: involves unauthorized **destruction** of data e.g. website defacement
- theft of service: involves unauthorized **use of resources** e.g. intruder installing a daemon on a system that acts as a file server.
- denial of service: involves **preventing legitimate use** of the system e.g. the original Internet worm that turned into a **DOS** attack. they can be accidental too.

- several methods are used by attackers: **masquerading** in which one participant in a communication pretends to be someone else. by masquerading they breach authentication, the correctness of identification, thus can gain access to information they are not normally allowed.
- another method is **replay attack** which is to replay a captured exchange of data. a replay attack consists of the malicious or fraudulent repeat of a valid data e.g. repeat of a request to transfer money. **man-in-the-middle attack** is another example.
- to protect a a system, we must take security measures at four levels: **physical, network, operating system, application.**

- **Program threats:**
- it is useful to log in to a system without authorization, very useful to leave behind a **remote access tool (RAT)** or back-door daemon provides information or allows easy access even if the original exploit is blocked.
- **malware**: is a software designed to exploit, disable or damage computer systems.
- many ways to perform. **trojan horse,** a program that acts in a malicious manner, rather than performing its stated function e.g. a flashlight app that accesses the user's contacts or photos and smuggles them to a remote server.
- **Viruses** is a fragment of code embedded in a legitimate program. They are self-replicating and are designed to “infect” other programs.
- UNIX and multiuser operating systems are susceptible to viruses because the executable programs are protected from writing by the operating systems. Viruses main categories: File, Boot, Macro, Rootkit, Source code, Polymorphic, Encrypted, Stealth, Multipartite and Armored.
- **System and Network Threats:**
- **Networks** are common and attractive targets, and hackers have many options for mounting network attacks.

- **Cryptography**
- The broadest tool to system designers and user is cryptography. An encryption algorithm consists of the following algorithms.:

    - a set K of keys
    - a set M of messages
    - a set C of ciphertexts
- Network protocols are typically organized in layers, with each layer acting as a client of the one below it.
- **Transport Layer Security (TLS)** is a cryptographic protocol that enables two computers to communicate securely— that is, so that each can limit the sender and receiver of messages to the other.

- **User Authentication:**
- **Passwords** are the most common approach to authenticating a user identity.
- **Password vulnerabilities:** they can be guessed in three common ways: (1) intruder (human or program) knows the user or have information about the user. (2) use brute force, try enumeration of all possible combination of valid password characters. (3) dictionary attacks where all words, word variations and common passwords are tried.
- **securing passwords:** The UNIX system uses **secure hashing** to avoid the necessity of keeping its passwords list secret. Because the password is hashed rather than encrypted, it is impossible for the system to decrypt the stored value and determine the original password. **salt** or a recorded random number is used in the hashing algorithm. The salt value is added to the password to ensure that if two plaintext passwords are the same, they result in different hash values.
- **one-time passwords:** to avoid the problems of sniffing and shoulder surfing, a system can use a set of **paired passwords.** when a session begins, the system randomly selects and presents one part of a password pair; the user must supply the other part.

PROTECTION

involves controlling the access of processes and users to the resources defined by a computer system.

- to provide this protection, we can use various mechanisms to ensure that only processes that have gained proper authorization from the operating system can operate on the files, memory segments, CPU, networking, and other resources of a system.

- **mechanisms** are distinct from **policies.** mechanisms determine **how** something will be done, policies decide **what** will be done.

Principles of protection:
- **principle of least privilege:** dictates that programs, users, and even systems be given just enough privileges to perform the tasks.
- **compartmentalization** is the process of protecting each individual system component through the use of specific permissions and access restrictions.

- **protection rings:**

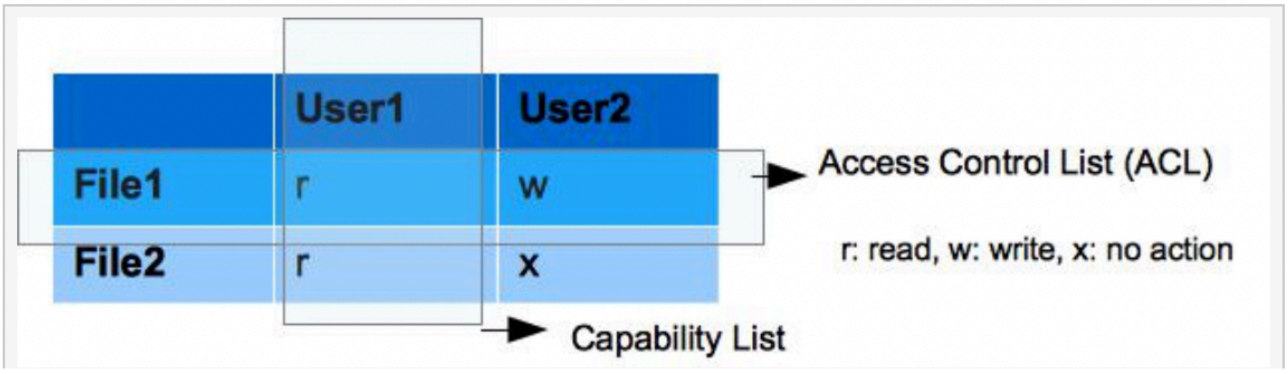
- **Access matrix:** the general model of protection can be viewed abstractly as a matrix, called an access matrix. the rows represent domains, and the columns represent objects.
- OS have traditionally used **discretionary access control (DAC)** for restricting access to files and other system objects. With DAC, access is controlled based on the identities of individual users or groups.
- **chgrp** command allows you to change the group ownership, and **chown** allows you to change both the user and group ownership.

- **Setuid**, which stands for set user ID on execution, is a special type of file permission in Unix and Unix-like operating systems such as Linux and BSD. It is a security tool that permits users to run certain programs with escalated privileges.
When an executable file's setuid permission is set, users may execute that program with a level of access that matches the user who owns the file. For instance, when a user wants to change their password, they run the passwd command. The passwd program is owned by the root account and marked as setuid, so the user is temporarily granted root access for that limited purpose.

- in UNIX, an owner identification and a domain bit, known as the **setuid bit,** are associated with each file.
- **Log file-** the activity of most services running in the system are written to a file inside this directory or one of its subdirectories.

-**Logrotate** is a system utility that manages the automatic rotation and compression of log files. If log files were not rotated, compressed, and periodically pruned, they could eventually consume all available disk space on a system.

- **Access control list (ACL)** is to specify user names and the types of access allowed for each user, this way it make access dependent on the identity of the user. If the user is listed for a requested access, the access is allowed; otherwise, a protection violation occurs thus the access is denied.
- We can look at Access matrix as a protection model for a system.
- The simplest implementation is a **global table** consisting of a set of ordered triples <domain, object, rights-set>. It has several drawbacks e.g. the table is usually large and thus cannot be kept in main memory, so additional I/O is required — although virtual memory techniques can be useful for managing the table.
- Each column in the access matrix can be implemented as an **access list** for one object. The resulting list for each object consists of ordered pairs <domain, rights-set>, which define all domains with a non-empty set of access rights for that object.
- A **capability list** for a domain is a list of objects together with the operations allowed on those objects. An object is often represented by its physical name or address, called a **capability.**
- An **access list** is a list for each object consisting of the domains with a nonempty set of access rights for that object. A **capability list** is a list of objects and the operations allowed on those objects for each domain.



	User1	User2
File1	r	w
File2	r	x

- **Race conditions** are involved in the event mechanism. If a process decides (because of checking a flag in memory, for instance) to sleep on an event, and the event occurs before the process can execute the primitive that does the sleep() on the event, the process sleeping may then sleep forever. We prevent this situation by raising the hardware processor priority during the critical section so that no interrupts can occur. Thus, only the process desiring the event can run until it is sleeping. Hardware processor priority is used in this manner to protect critical regions throughout the kernel and is the greatest obstacle to porting UNIX to multiple-processor machines. However, this problem has not prevented such porting from being done repeatedly.

- The way of communicating a message from one process to another process is called a **signal**.

- Linux divides the privileges traditionally associated with superuser into distinct units, known as **capabilities**, which can be independently enabled and disabled. Capabilities are a per-thread attribute.
- A capability table is bound to a subject, whereas an ACL is bound to an object.

- **Rootkit** viruses are originally the back-doors on UNIX systems meant to provide easy root access. When malware infects the operating system, it can take over all of the system's functions, including those functions that would normally facilitate its own detection.

-