Operating Systems Security Considerations

Mohammad Heidari
Security.Papers@Gmail.com

Abstract

An operating system is a fundamental component of most modern computer systems, Operating systems are close to the hardware, it masks the details of the underlying hardware from the programmer and provides lowest layer of software visible to users. It can be viewed as a resource manager, responsible for fair resource sharing between different processes in the system. On the other hand, Operating systems control access to application memory and scheduling of the processor. Applications must be run like OS-Level services and the developer of these apps does not know the level of details needed to develop secure applications on their own, If the OS isn't doing these things securely, it generally compromise all security at higher levels, So The OS is a very logical place to enforce and support security. This paper presents the common built-in security criteria and mechanisms in OSs, it also introduces the common Vulnerabilities .

Keywords: Operating system security, Memory protection, Access Control, Vulnerability

Introduction

OS Security revolves around the appropriate protection of four elements. **Confidentiality** prevents or minimizes unauthorized access and disclosure of data and information. **Integrity** makes sure that the data being worked with is actually the correct data. **Availability** (as defined in RFC 2828) is the property of a system or system resource being accessible and usable upon demand by an authorized system entity, according to performance specification for the system. **Authenticity** makes possible that a computer system be able to verify the identity of a user.
Availability deals with the computer system assets (Hardware, Software, and Data). Hardware is the most vulnerable to attack (Accidental and deliberate damage to equipment as well as theft). A key threat to software is an attack on availability (Configuration management, Software modification, and so on). However the discussion about availability is beyond the scope of this paper. Confidentiality and Integrity deal with the three important roles: Protection Models, Capability and Assurance. One of the important factors of Confidentiality and integrity is Protection models. The model is the most important aspect of security, even if everything else in the system is perfect, it will still be exploitable if a weak model is used. Each of the protection models should be proven to ensure that they are as close as possible .Well known protection models include the Bell-LaPadula hierarchical mandatory access confidentiality model and the Biba hierarchical integrity model. Systems may tend to use several protection models rather than a single comprehensive one. The next part is Capability. Capabilities are the tools and functionality that the operating system uses to implement a given model and may include things like the specific access controls or what privileges are available and how they are defined. Examples include groups, how setting the system time is controlled, or having the system crash when it is unable to audit particular events. The last part of Confidentiality and integrity is Assurance. Assurances are a way of determining that the models are implemented correctly and cannot be bypassed, additionally assurances can cover nearly all aspects of the operating system, from the maturity level of the development team to the quality and comprehensiveness of the documentation to the architecture of the operating system itself. For example, using microkernel architecture allows for much higher assurances as all aspects of the protection models may be implemented at a single point known as a reference monitor.

Protection Mechanisms

The concept of multiprogramming introduces the sharing resources among users. This sharing involves Memory, I/O devices, Programs and Data. The ability to share these resources introduces the need for protection. An OS may offer protection along the following Spectrum: [1]

No Protection: This is appropriate when sensitive procedures are being run at separate times.

Isolation: This approach implies that each process operates separately from other processes, with no sharing. Each process has its own address space, files, and other objects

Share all or Share nothing: In this method, the owner of an object declares it to be public or private, in the other words, only the owner‘s processes may access the object.

Share via access limitation: The OS checks the permissibility of each access by a specific user to specific object; the OS therefore acts as a guard between users and objects, ensuring that only authorized accesses occur.

Share via dynamic capabilities: This extends the concept of access control to allow dynamic creation of sharing rights for objects.

Limit use of an object: This form of protection limits not just access to an object but the use to which that object may be put.

A given OS may provide different degree of protection for different objects, users and applications The OS needs to balance the need to allow sharing, with the need to protect the resources of individual users.

Protection of Memory

In a multiprogramming environment, protection of main memory is essential. The concern here is not just security but the correct functioning of the various processes that are active. The separation Of the memory space of various processes is easily accomplished with a virtual-memory scheme. Segmentation or Paging, or two in combination, provides an effective tools of managing main memory. If complete isolation is sought, then the OS must simply ensure that each segment or page accessible only by the process to which it is assigned. This is accomplished by requiring that there be no duplicate entries in page and/or segment tables. If sharing is to be allowed then the same segment or page may appear in more than one table. Segmentation specially lends itself to the implementation of protection and sharing policies. Because each segment table entry includes a length as well as a base address. A program can not access a main memory location beyond the limit of a segment. To achieve sharing, it is possible for a segment to be referenced in the segment tables of more than on process. In the paging system, the page structure of the programs and data is not visible to the programmer.

The measures taken to control access in a data processing systems fall into two categories: User-Oriented and Data-Oriented

User-Oriented Access Control

User control of access is sometimes referred to as Authentication. [2] the most common technique for user access control on a shared system or server is the user log, which requires ID and Password. User access control in distributed environment can be either centralized or decentralized In a centralized approach network provides a log on service, determining who is allowed to use the network and to whom the user is allowed to connect. Decentralized user access control treats the network as a transport communication link, and the destination host carries out the usual log on procedure. In many networks, two levels of access control may be used.

Data-Oriented Access Control

Following successful log on , the user has been granted access to one or set of hosts and applications. At this time we need Data access control. In this regard real world operating system protection models fall basically into one of two types:
1- **Mandatory access controls (MAC) 2- Discretionary access controls (DAC)**

In computer security passive resources are called ***objects*** and active entities that utilize the resources are called ***subjects.*** Typical objects include: files, directories, memory, printers … And typical subjects include: users, processes… The roles depend on situation: For example, a Process can request access to some resource (act as a subject) and later be a target of access request (act as an object)

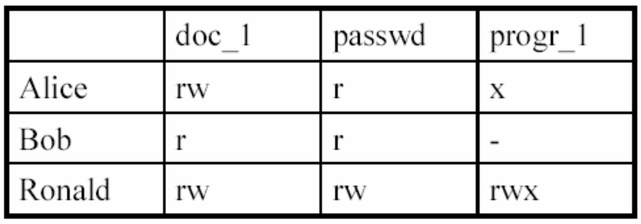
Mandatory access controls (MAC)

In Mandatory access controls, also called multilevel access control, Objects (information) are classified on hierarchical levels of security sensitivity (typically, top secrets, secret, confidential) Subjects (Users) are assigned their security clearance. Access of a subject to an object is granted or denied depending on the relation between the clearance of the subject and the security classification of the object. Lattice model and Bell-LaPadula model are based on MAC

Discretionary access controls (DAC)

Each object has its unique owner. The owner exercises its discretion over the assignment of access permissions. Lampson introduced the access matrix model for DAC. The core of this model is a matrix whose rows are indexed by subjects and columns by objects.

Figure 1
Access Matrix



	doc_1	passwd	progr_1
Alice	rw	r	x
Bob	r	r	-
Ronald	rw	rw	rwx

In real systems, however, access control matrices are not very practical, because the matrix is usually sparse and there is a lot of redundancy and new subjects and objects can be added or removed easily, but the centralized matrix could become a bottleneck. The matrix may be decomposed by columns, yielding Access Control List (ACL). (Figure 2) Thus for each object, an ACL details users and their permitted access rights. ACL may contain a default or public entry. Decomposition by rows yields capability tickets (Figure 3). A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to lend or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than ACL. To accomplish this problem, OS hold all tickets on behalf of the users. These tickets would have to be held in a region of memory inaccessible to users.

Figure 2
ACL

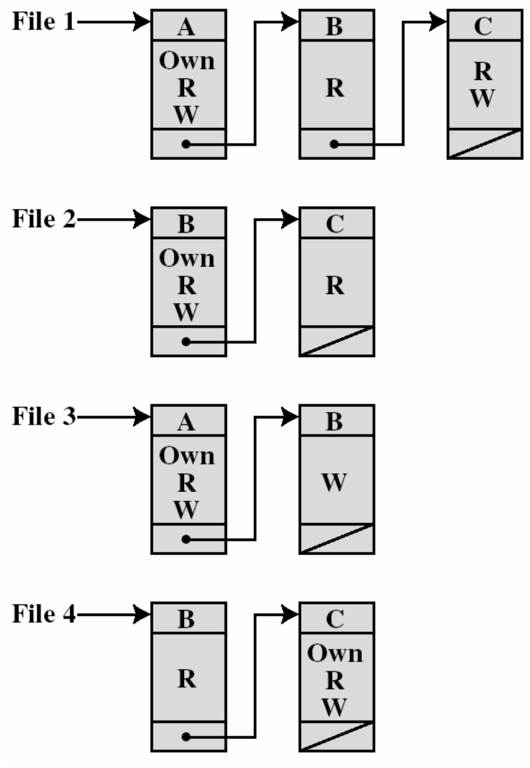
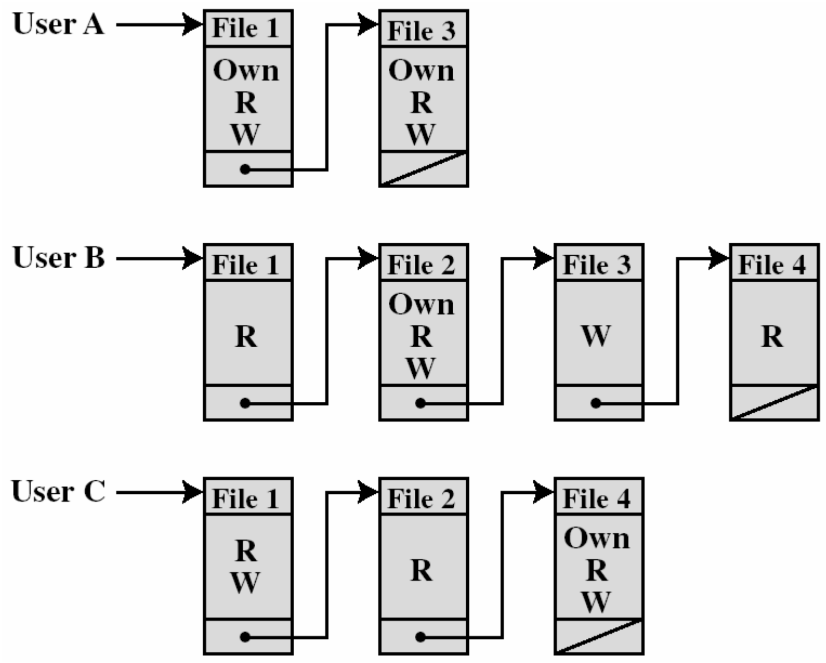


Figure 3
Capability lists



Protection Based on an OS mode

Most processor supports at least two modes of operations: 1- User mode 2- Kernel mode. The reason for using two modes should be clear. It is necessary to protect the OS and the key OS table such as process control blocks, from interference by user programs in kernel mode. This level of control is not necessary for user mode.

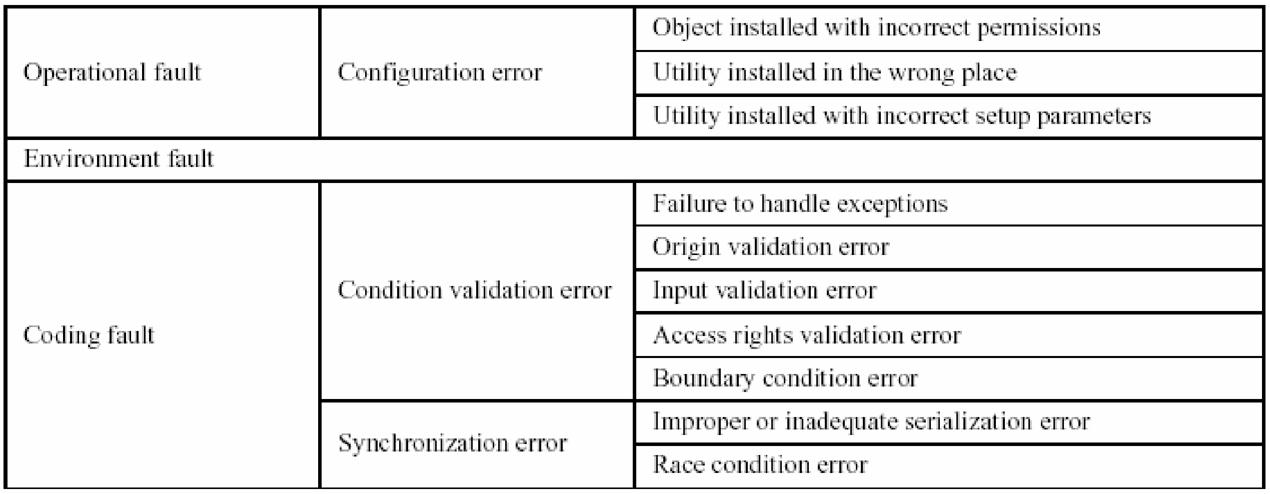
File Sharing

Multi user systems almost always require that files can be shared among a number of users. We have several access rights such as Reading, Appending, Updating, Changing protection… these Access rights can be provided to different classes of users such as Specific user (with user ID), user groups and all users. When access is granted to more than one user to append or update a file The OS must enforce discipline. A brute-force approach is to allow a user to lock the entire file when it is to be updated.

Operating Systems Vulnerabilities

Vulnerabilities will probably always exist in large and complex software systems. At least with today's software methods, techniques, and tools, it seems to be impossible to completely eliminate all flaws. Operating systems are examples of software components that are large, very complex, and vulnerable. At the same time, these components play an important role in the achievement of overall system security, since many protection mechanisms and facilities, such as authentication and access control, are provided by the operating system. Vulnerability is defined as a place at which the probability of a breach exceeds a predefined threshold. You can see one of the taxonomy that was presented in Categorization of Security Vulnerabilities in the figure 4 [3]

Figure 4
Taxonomy of security faults



Operational fault	Configuration error	Object installed with incorrect permissions
		Utility installed in the wrong place
		Utility installed with incorrect setup parameters
Environment fault		
Coding fault	Condition validation error	Failure to handle exceptions
		Origin validation error
		Input validation error
		Access rights validation error
		Boundary condition error
	Synchronization error	Improper or inadequate serialization error
		Race condition error

Observations on common operating system vulnerabilities indicate that similar vulnerabilities have been grouped together in a heuristic manner rather than according to some formal model. A collection of five common security problems has been identified and will be further described below. These are:

• Improper input validation
• Weak cryptographic algorithms
• Weak authentication protocols
• Insecure bootstrapping
• Configuration mistakes

The first four of these have a technical or system-related basis, while the latter is related to Organizational problems or management. This implies that they are not orthogonal, specific vulnerability may belong to more than one group. Not all vulnerabilities originate from the operating system itself. A badly chosen password, for example, is not an operating system vulnerability, but rather a user related weakness, caused by the fact that the user is either unaware of the possible consequences or just careless.

Improper input validation: In general, it is essential to carefully check the input to software routines, i.e., to perform an input validation. The check may be with regard to the number of parameters provided, the type of each parameter, or to simply ensure that the amount of input data is not larger than the buffer allocated to store the data. Improper or non-existent input validation is a well-known and serious problem in operating systems.

Weak cryptographic algorithms: Overly weak algorithms in cryptographic systems are, in our opinion, another security problem. In operating systems, cryptographic algorithms have long been used to encrypt passwords. However, if the algorithm in use is not sufficiently strong, an attacker may manage to derive plaintext passwords from its corresponding encrypted representation. Note that a strong cryptographic algorithm can suddenly become weak as a result of a research break through in, for example, number theory.

Weak authentication protocols: Before a user gains permissions to access resources, he must prove his identity. This process is called authentication. Most authentication systems are based on a secret common to the parties involved. The authentication mechanism is most often simply a password, a secret word known to the system and the user only. However, accomplishing a secure authentication procedure is a complex task, especially in a distributed environment.

Insecure bootstrapping: From our security analyses it is apparent that system initialization is a major security problem in today's operating systems. All studied systems were vulnerable during bootstrapping. For example, many attackers discovered that the SunOS was easily rebooted in single-user mode. Commands entered in that mode run with root privileges, and it was possible to extend these privileges to the server, or. A Windows NT system executing on a PC can most often be rebooted with a foreign operating system, such as MS-DOS. Once a foreign system is booted on the PC, the NTFS volume may be mounted. Access to files on the newly mounted volume will bypass the access control mechanism enforced when Windows NT is operating.

Configuration mistakes: In current operating systems, security features and mechanisms are seldom activated by default. A secure “out-of-the-box” installation is the exception rather than the rule. To achieve an acceptable security level, the system owner must, after he has completed installation, spend quite some time in securing the system. However, securing a system is not a trivial task, since operating systems are large and complex. In addition, the number of skilled practitioners in computer security is very small.

Malicious codes are important threats (such as Trojan Horses…) that are mentioned in “Malicious Codes in Depth” [4]

Conclusion

This paper explains main aspects in OS security. Much of the work in security and protection as it relates to OSs can be roughly grouped into three elements:

1- **Access Control:** Concerned with regulating user access to the total system, subsystems, and regulating process access to various resources and objects within the system.
2- **Information Flow Control:** regulates the flow of data within the system and it’s delivery to users.
3- **Certification:** relates to proving that access and flow control mechanism perform according to their specifications and that they enforce desired protection and security policies.

References

[1] Pfleeger, c. Security in computing upper saddle River, NJ: Prentice-Hall PTR, 1997
[2] William Stallings “Security “, school of Info technology, Griffith University, 2001
[3] C.R. Attanasio, P. Markstein and R.J Philips, Penetrating on OS, IBM system Journal, 1996
[4] Mohammad Heidari, “Malicious Codes in Depth”, Securitydocs.com, 2004