ISSN: 2321-7782 (Online)
Volume 3, Issue 3, March 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A Preliminary Review on Trusted Operating System

Rani Shriram Joshi[1]
Computer Science and Engineering
H.V.P.M's C.O.E.T
Amravati, India

Dr. Anjali.B. Raut[2]
Computer Science and Engineering
H.V.P.M's C.O.E.T
Amravati, India

Abstract: ***Operating systems are at the heart of security systems for modern computers. Operating systems must provide mechanisms for both separation and sharing, mechanisms that must be robust and yet easy to use. Developing secure operating systems involves four activities. First, the environment to be protected. Through policy statements and models, the essential Components of systems are identified, and the interactions among components. Trusted Operating Systems has presented a variety of policies and models of security. Whereas the policies covered confidentiality and integrity, the models ranged from reference monitors and information flow filters to multilevel security and integrity models. Models such as that of Bell and La Padula describe permissible access in a multilevel environment, and the HRU model demonstrates the limits of computer security.***

Keywords: ***Trusted System, Trusted Software, Security Features, Reference Monitor, Trusted Computing Base (TCB)***

I. INTRODUCTION

a) TRUSTED System

A Trusted system is a system that is relied upon to a specified extent to enforce a specified security policy. As such, a trusted system is one whose failure may break specified security policy .Four major underpinnings of a trusted operating system:

Policy: Every system can be described by its requirements: statements of what the system should do and how it should do it. An operating system's security requirements are a set of well-defined, consistent, and implementable rules that have been clearly and unambiguously expressed. If the operating system is implemented to meet these requirements, it meets the user's expectations.

Model: To create a trusted operating system, the designers must be confident that the proposed system will meet its requirements while protecting appropriate objects and relationships. They usually begin by constructing a model of the environment to be secured. The model is actually a representation of the policy the operating system will enforce. Designers compare the model with the system requirements to make sure that the overall system functions are not compromised or degraded by the security needs.

Design: After having selected a security model, designers choose a means to implement it. Thus, the design involves both what the trusted operating system is (that is, its intended functionality) and how it is to be constructed (its implementation).

Trust: Because the operating system plays a central role in enforcing security, we (as developers and users) seek some basis (assurance) for believing that it will meet our expectations. Our trust in the system is rooted in two aspects: *features* (the operating system has all the necessary functionality needed to enforce the expected security policy) and *assurance* (the operating system has been implemented in such a way that we have confidence it will enforce the security policy correctly and effectively).

b) TRUSTED SOFTWARE

Trusted software is often used as a safe way for general users to access sensitive data. Trusted programs are used to perform limited (safe) operations for users without allowing the users to have direct access to sensitive data.

The software is trusted software if we know that the code has been rigorously developed and analyzed, giving us reason to trust that the code does what it is expected to do and nothing more. Typically, trusted code can be a foundation on which other, untrusted, code runs. That is, the untrusted system's quality depends, in part, on the trusted code; the trusted code establishes the baseline for security of the overall system. In particular, an operating system can be trusted software when there is a basis for trusting that it correctly controls the accesses of components or systems run from it. For example, the operating system might be expected to limit users' accesses to certain files.

To trust any program, we base our trust on rigorous analysis and testing, looking for certain key characteristics:

1. *Functional correctness:* The program does what it is supposed to, and it works correctly.
2. *Enforcement of integrity:* Even if presented erroneous commands or commands from unauthorized users, the program maintains the correctness of the data with which it has contact.
3. *Limited privilege:* The program is allowed to access secure data, but the access is minimized and neither the access rights nor the data are passed along to other untrusted programs or back to an untrusted caller.
4. *Appropriate confidence level: The* program has been examined and rated at a degree of trust appropriate for the kind of data and environment in which it is to be used.

II. LITERATURE REVIEW AND RELATED WORK

In October 1967, a task force was assembled under the auspices of the Defense Science Board to address computer security safeguards that would protect classified information in remote-access, resource-sharing computer systems. The Task Force report, "Security Controls for Computer Systems," published in February 1970, made a number of policy and technical recommendations on actions to be taken to reduce the threat of compromise of classified information processed on remote-access computer systems. In September 2005, IBM entered CCEVS evaluation with trusted Linux for the label security protection platform, role-based access control profile, and the control led access protection profile at the EAL4 level. This evaluation is being conducted across IBM's System server line of products. Evaluation is expected to be completed in the first quarter of 2007. IDC's Government In sights believes not only that trusted operating system will give SIs another powerful tool in their system development arsenal but also that trusted operating system has the potential to become one of the more popular operating systems installed for government solutions that will be built by man Sis.

III. SECURITY FEATURES OF TRUSTED OPERATING SYSTEM OVER ORDINARY OPERATING SYSTEM

Unlike regular operating systems, trusted systems incorporate technology to address both features and assurance. The design of a trusted system is delicate, involving selection of an appropriate and consistent set of features together with an appropriate degree of assurance that the features have been assembled and implemented correctly. Figure 2 illustrates how a trusted operating system differs from an ordinary one. Compare it with Figure 1. Notice how objects are accompanied or surrounded by an access control mechanism, offering far more protection and separation than does a conventional operating system. In addition, memory is separated by user, and data and program libraries have controlled sharing and separation.

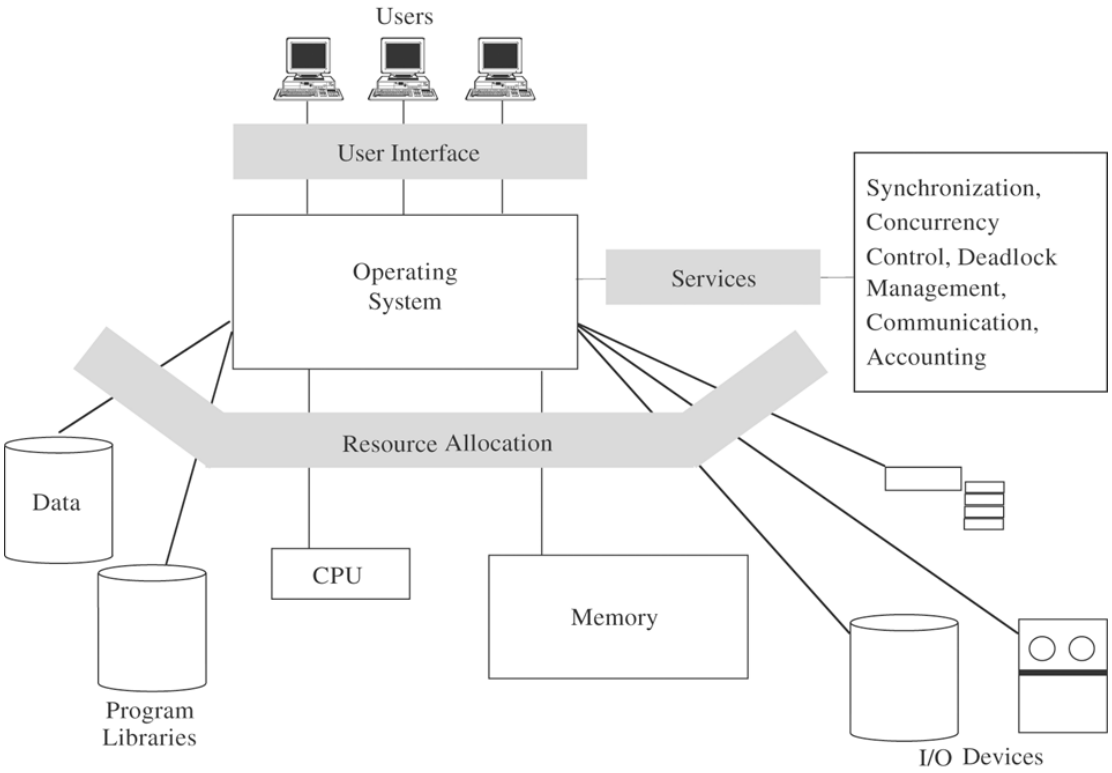


Figure 1. Overview of an Operating System's Functions.

The key features of a trusted operating system security:

a) Identification and Authentication

Identification is at the root of much of computer security. We must be able to tell who is requesting access to an object, and we must be able to verify the subject's identity. As we see shortly, most access control, whether mandatory or discretionary, is based on accurate identification. Identification involves two steps: finding out who the access requester is and verifying that the requester is indeed who he/she/it claims to be. That is, we want to establish an identity and then authenticate or verify that identity. Trusted operating systems require secure identification of individuals, and each individual must be uniquely identified

b) Mandatory and Discretionary Access Control

Mandatory access control (MAC) means that access control policy decisions are made beyond the control of the individual owner of an object. A central authority determines what information is to be accessible by whom, and the user cannot change access rights. An example of MAC occurs in military security, where an individual data owner does not decide who has a top-secret clearance; neither can the owner change the classification of an object from top secret to secret.

c) *Object Reuse Protection*

One way that a computing system maintains its efficiency is to reuse objects. The operating system controls resource allocation, and as a resource is freed for use by other users or programs, the operating system permits the next user or program to access the resource. But reusable objects must be carefully controlled, lest they create a serious vulnerability. To see why, consider what happens when a new file is created. Usually, space for the file comes from a pool of freed, previously used space on a disk or other storage device. Released space is returned to the pool "dirty," that is, still containing the data from the previous user. Because most users would write to a file before trying to read from it, the new user's data obliterate the previous owner's, so there is no inappropriate disclosure of the previous user's information. However, a malicious user may claim a large amount of disk space and then scavenge for sensitive data. This kind of attack is called object reuse. The problem is not limited

to disk; it can occur with main memory, processor registers and storage, other magnetic media (such as disks and tapes), or any other reusable storage medium.

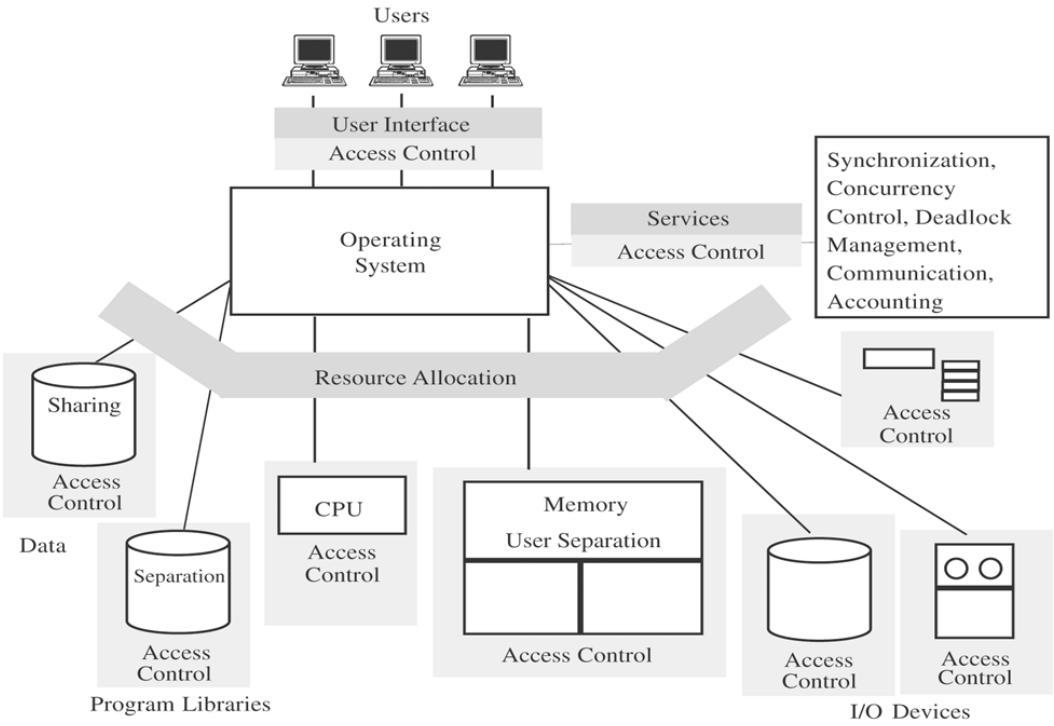


Figure 2. Security Functions of a Trusted Operating System.

d) Complete Mediation

For mandatory or discretionary access control to be effective, *all* accesses must be controlled. It is insufficient to control access only to files if the attack will acquire access through memory or an outside port or a network or a covert channel. The design and implementation difficulty of a trusted operating system rises significantly as more paths for access must be controlled. Highly trusted operating systems perform complete mediation, meaning that all accesses are checked.

e) *Trusted Path*

One way for a malicious user to gain inappropriate access is to "spoof" users, making them think they are communicating with a legitimate security enforcement system when in fact their keystrokes and commands are being intercepted and analyzed. For example, a malicious spoofer may place a phony user ID and password system between the user and the legitimate system.

As the illegal system queries the user for identification information, the spoofer captures the real user ID and password; the spoofer can use these bona fide entry data to access the system later on, probably with malicious intent. Thus, for critical operations such as setting a password or changing access permissions, users want an unmistakable communication, called a trusted path, to ensure that they are supplying protected information only to a legitimate receiver. On some trusted systems, the user invokes a trusted path by pressing a unique key sequence that, by design, is intercepted directly by the security enforcement software; on other trusted systems, security relevant changes can be made only at system startup, before any processes other than the security enforcement code run.

f) Accountability and Audit

A security-relevant action may be as simple as an individual access to an object, such as a file, or it may be as major as a change to the central access control database affecting all subsequent accesses. Accountability usually entails maintaining a log of security-relevant events that have occurred, listing each event and the person responsible for the addition, deletion, or change. This audit log must obviously be protected from outsiders, and every security-relevant event must be recorded.

g) Intrusion Detection

Intrusion detection software builds patterns of normal system usage, triggering an alarm any time the usage seems abnormal. After a decade of promising research results in intrusion detection, products are now commercially available. Some trusted operating systems include a primitive degree of intrusion detection software.

IV. TRUSTED OPERATING SYSTEM DESIGN

a) TRUSTED SYSTEM DESIGN ELEMENTS

Good design principles are always good for security. But several important design principles are quite particular to security and essential for building a solid, trusted operating system. These principles have been articulated well by Saltzer [SAL74] and Saltzer and Schroeder [SAL75]:

1. *Least privilege.* Each user and each program should operate by using the fewest privileges possible. In this way, the damage from an inadvertent or malicious attack is minimized.
2. *Economy of mechanism.* The design of the protection system should be small, simple, and straightforward. Such a protection system can be carefully analyzed, exhaustively tested, perhaps verified, and relied on.
3. *Open design.* The protection mechanism must not depend on the ignorance of potential attackers; the mechanism should be public, depending on secrecy of relatively few key items, such as a password table. An open design is also available for extensive public scrutiny, thereby providing independent confirmation of the design security.
4. *Complete mediation.* Every access attempt must be checked. Both direct access attempts (requests) and attempts to circumvent the access checking mechanism should be considered, and the mechanism should be positioned so that it cannot be circumvented.
5. *Permission based.* The default condition should be denial of access. A conservative designer identifies the items that *should* be accessible, rather than those that should *not.*
6. *Separation of privilege.* Ideally, access to objects should depend on more than one condition, such as user authentication plus a cryptographic key. In this way, someone who defeats one protection system will not have complete access.
7. *Least common mechanism.* Shared objects provide potential channels for information flow. Systems employing physical or logical separation reduce the risk from sharing.
8. *Ease of use.* If a protection mechanism is easy to use, it is unlikely to be avoided.

Although these design principles were suggested several decades ago, they are as accurate now as they were when originally written. The principles have been used repeatedly and successfully in the design and implementation of numerous trusted systems. More importantly, when security problems have been found in operating systems in the past, they almost always derive from failure to abide by one or more of these principles. At the center of trusted operating system, there is a security kernel. Gollman described the concept of security kernel in the terms of the Trusted Computer System Evaluation Criteria that have been used as the evaluation guideline of trusted operating systems.

The related terms can be summarized as follows.

1. The trusted operating system is the implementation of Trusted Computing Base.
2. The trusted computing base (TCB) is the totality of computer system's protection mechanisms and it enforces a unified security Policy to guarantee confidentiality, integrity, and availability of the system [2,3,4].

TCB Functions are:

1. Hardware, including processors, memory, registers, and I/O devices
2. Some notion of processes, so that we can separate and protect security-critical processes
3. Primitive files, such as the security access control database and identification/authentication data
4. Protected memory, so that the reference monitor can be protected against tampering
5. Some interprocess communication, so that different parts of the TCB can pass data to and activate other parts. For example, the reference monitor can invoke and pass data securely to the audit routine.

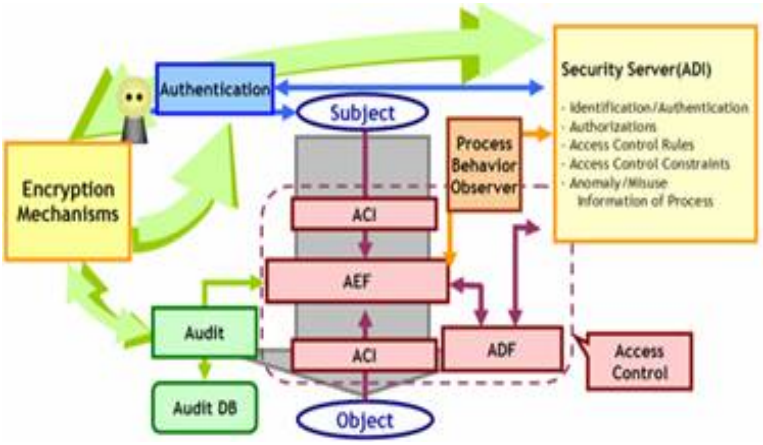


Figure 3: Trusted OS Architecture (Themis Kernel)

The *security kernel* is the software and hardware elements that enforce access control. The security kernel mediates all accesses of a system, and then only legitimate accesses are performed in the system. The security kernel of trusted operating systems implements the reference-monitoring concept of the trusted computing base.

The *reference monitor* is a conceptual model of access controls that mediates all accesses in a system a reference monitor must be:

1. *tamperproof*, that is, impossible to weaken or disable
2. *unbypassable*, that is, always invoked when access to any object is required
3. *analyzable*, that is, small enough to be subjected to analysis and testing, the completeness of which can be ensured

A reference monitor can control access effectively only if it cannot be modified or circumvented by a rogue process, and it is the single point through which all access requests must pass.

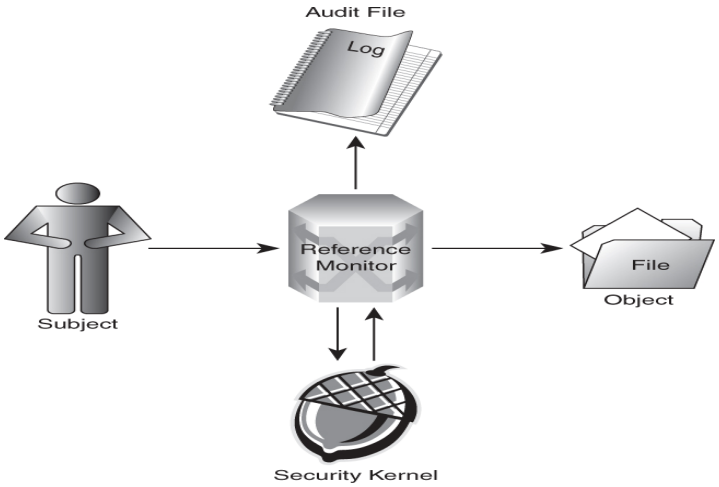


Figure 4. Reference Monitor

V. Conclusion

Trusted operating systems are more secure that the traditional model of operating system security. Operating systems by themselves (regardless of their security constraints) are very difficult to design. Trusted operating systems handle many duties, are subject to interruptions and context switches, and must minimize overhead so as not to slow user computations and interactions.

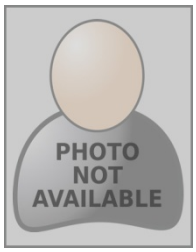
Acknowledgment

My thanks to the Guide, Dr.A.B.Raut and Principal Dr.A.B.Marathe, who provided me constructive and positive feedback during the preparation of this paper.

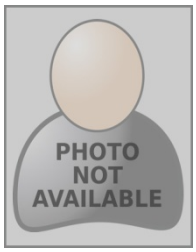
References

1. C.P.Pfleeger and S.L.Pfleeger, "Security in Computing ", PEARSON Fourth Edition
2. D.Gollmann, Computer Security, John Wiley & SONS, 1999.
3. United States Government National Computer Security Council, Trusted Computer System Evaluation Criteria", Department of Defense Standard(DOD 5200.28-STD), Library Number S225, 711, 1985.
4. E. G. Amoroso, Fundamentals of Computer Security Technology, AT\&T Bell Laboratories, Prentice Hall PTR., 1994.
5. T. Ptacek and T. Newsham, Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, 1998.

Author(s) Profile



Rani Shriram Joshi, received the B.E.degree in Information Technology from H.V.P.M's College Of Engineering And Technology, Amravati in 2014. She is currently pursuing Master's Degree in Computer Science and Engineering from H.V.P.M's College of Engineering And Technology, Amravati.



Dr. Anjali.B. Raut, received the B.E.and M.E degree in Computer Science from Prof. Ram Meghe Institute of Technology, Badnera and PHD from Sant Gadge Baba Amravati University. She is currently working as Head Of Department in Computer Science and Engineering at H.V.P.M's college of Engineering and Technology, Amravati.

