

APPLYING THE TCSEC GUIDELINES TO A REAL-TIME EMBEDDED SYSTEM ENVIRONMENT*

Jim Alves-Foss, Deborah Frincke and Gene Saghi
Laboratory for Applied Logic
Department of Computer Science
University of Idaho
Moscow, ID 83844-1010

Abstract

The DoD Trusted Computer System Evaluation Criteria (TCSEC) was developed to provide a common yardstick for evaluating system security, a guide for system developers, and as a procurement standard. Since these guidelines were released, it has become important to consider the security of systems other than the traditional operating systems that influenced the TCSEC. Multilevel data security is required of many modern advanced, real-time embedded systems. In this paper, we discuss real-time embedded systems such as those found in avionics systems and how the TCSEC's requirements may be modified to suit such systems.

Introduction

The DoD Trusted Computer System Evaluation Criteria (TCSEC) [1] was developed to provide a common yardstick for evaluating system security, as a guide for system developers, and as a procurement standard. However, since then it has become important to consider the security of systems other than the traditional operating systems that influenced the TCSEC's development. Modern real-time systems used in all types of manufacturing lines may require security. As these systems become more and more integrated and as the amount of data sharing increases, security and validation will become increasingly important. Industrial spying is becoming a very serious problem in some industries.

Real-time systems are becoming quite complex, and may handle sensitive tasks. The F-22 avionics system features a high-performance, shared-memory, heterogeneous multiprocessor connected to sensors and instrumentation by high-bandwidth fiber-optic interconnects. The F-22 operating system supports dynamic assignment of tasks to processing assets (data processors and signal processors). The Boeing 777 has over 2.6 million lines of code in the avionics and cabin-entertainment system. The heart of the system, Airplane Information Management System (AIMS) built by Honeywell Air

*The research was funded in part from a grant by Texas Instruments.

Transport Systems Inc., handles flight management, cockpit displays, and central maintenance, and modules share processors, memory system, and operating system. Software for AIMS alone is over 600,000 lines of code [10].

One definition of a secure system requires that it protect the information it stores from unauthorized release or modification. The Multilevel Security Policy (MLS) as described in the TCSEC (for B1 and above) associates security levels with subjects (e.g., program, user) and objects (e.g., data sets, memory), and requires that the contents of objects can only be seen by a subject at its level or lower; that is, information can flow to the same or higher levels but never to lower levels. This mandatory security policy is augmented by a discretionary policy that further restricts information on a need-to-know basis. More abstract and general models of security that avoid the need to consider objects have been formulated by Goguen and Meseguer [4, 5], Fiertag, Levitt and Robinson [2], McCullough [8, 9] and McClean [6, 7]. In these models, the information a subject observes is dependent on the actions of subjects at the same level or lower. That is, the actions of higher-level subjects cannot be observed by lower-level subjects.

In a computer system the burden of security usually falls mostly on the operating system. An operating system that satisfies the MLS policy must enforce access control; it must not permit processes to have access to objects in violation of the security policy. In addition, the operating system itself must not be a channel for the communication of information not in accordance with the security policy. Such unwanted information flow can potentially occur through objects managed by the operating system and shared by more than one subject, or through timed performance of actions on shared resources. The term *covert channel* is often used in referring to such objects. There have been successful attempts to develop systems that implement the MLS policy, mostly for single host/multiple user systems such as mainframes or shared workstations. Regardless of the policy or model used to develop the system, there is the requirement to provide assurance that the implementation satisfies that model. The TCSEC specifies the types of assurance required to meet various levels of security certification. The assurance may consist of informal arguments, test documentation, formal models and descriptions and formal verification.

Background

Distributed and Real-Time Systems

For purposes of this paper, a distributed system is considered to consist of hosts (e.g., control systems, data acquisition systems, data analysis, and user interfaces), servers (e.g., repositories for objects accessible to multiple hosts, such as files, directories, names, data sets, shared memory), and a network through which the hosts and servers communicate. Security is especially important for distributed systems since such systems often have hosts and servers with different security classifications and certification levels, some with no access control (untrusted) and others that are multi-level secure (MLS). This is especially true of an open-system architecture with components supplied from different vendors.

Architectures for MLS distributed systems vary according to the services offered by the system. In a simple case, each host can support a single user or, more generally, several users operating at the same level. Here the burden of assuring security can fall on the network, which can mediate all communication between hosts to ensure only those intended to communicate with each other do so. [11] Indeed, since users are permitted to communicate only through a few well-defined interfaces,

it is easier to show compliance with the security policy for this distributed system than for most common multiuser mainframes. A more general distributed system would support multilevel hosts. Some of these systems permit the sharing of services or hosts across the system, perhaps through process migration. Here, one must prove the hosts secure in addition to requiring trusted interhost services.

The focus of this paper specifically involves distributed real-time embedded syducing the overhead required for maintaining system security. For the purposes of this paper, a real-time system is one which provides mechanisms to ensure that executing system tasks will meet specific performance and deadline criteria. An embedded system is one which is used specifically to monitor and control attached peripherals (in our example, the peripheral are sensor and weapons systems on a fighter aircraft). Such an embedded real-time system has a very limited user interface and can not be considered a general-purpose multi-user system. The addition of security features to such systems is often seen as an added processing burden that is unrealizable in a real-time system (or at least cost prohibitive). The paper identifies those aspects of the TCSEC which are not absolutely necessary for such systems, thereby reducing the overhead required for maintaining system security.

The TCSEC Guidelines

The TCSEC provides metrics against which systems can be evaluated, guidance for system development, and procurement guidelines. These guidelines specify both system properties and assurance requirements. The system properties are specific to a system configuration and are geared for a multi-user general purpose operating system. The assurance requirements are required for any type of system, regardless of operational environment or design.

The TCSEC's four divisions of certification are lettered A to D, where division A is the highest classification and division D is the lowest. Division D, or *minimal protection* systems have been evaluated but fail to meet requirements of a higher classification. Division C systems have provisions for *discretionary access control* (protection under the user's control) and *audit*. Division B systems additionally provide *mandatory access control*, or must implement mandatory restrictions on information flow between different security levels. The more restrictive subdivisions in this division require a formal statement of the security policy, documentation of the system design, testing to assure that the design is consistent with the specification, analysis of covert storage and timing channels, and permit only security relevant code in the reference monitor¹. Finally, Division A or *verified protection* systems have the same functionality as required for B3 systems, and developers must provide additional assurance that the system design correctly reflects the specification and implements the security policy. Assurance is gained through the use of formal design specifications and formal verification of these specifications.

Within each division of certification is a set of requirements, which specify behavior, design and operation of the computing system. Requirements are divided into one of four categories: security policy, accountability, assurance, and documentation. The security policy provides guidelines for the evaluation of discretionary access controls, object reuse, labeling, import and export of labeled objects, and mandatory access controls. Accountability provides guidelines for the evaluation of identification and authentication mechanisms, auditing, and trusted path access. Assurance provides guidelines for the evaluation of the system life cycle including system design, testing, analysis, management and maintenance. Documentation provides guidelines for the evaluation of the system documenta-

¹The TCSEC document defines a reference monitor as a system task that manages all references and validates them according to the security policy.

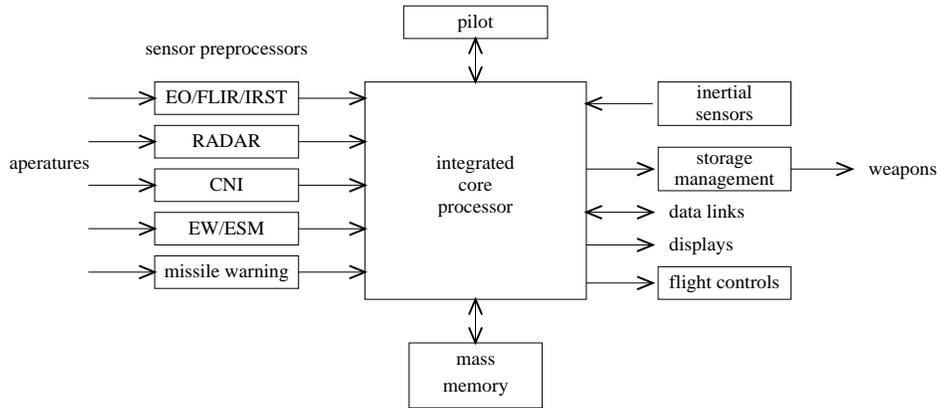


Figure 1: Block diagram of an avionics system.

tion, both from the user and management perspective as well as from the design and maintenance perspective.

Security Guidelines in a Real-Time Environment

The TCSEC was designed for evaluating multi-user, multi-level (for levels B1 and greater), general-purpose computing systems. Certain real-time embedded systems differ enough from these traditional systems to such an extent that many of the original TCSEC standards do not apply. We present aspects of real-time systems which affect the applicability of the TCSEC guidelines, emphasizing category B3, which incorporates all lower level functionality and differs from A1 only in the amount of assurance required. In the following subsections, we discuss a real-time embedded avionics system, enumerate differences between traditional TCSEC systems and our target real-time system, and analyze aspects of the TCSEC guidelines as they apply to real-time embedded avionics systems. Our discussion is summarized in Table 1.

An Avionics Real-Time Embedded Computer System

Modern real-time computer systems are migrating from purely proprietary architectures to open system architectures. It was once thought that only proprietary architectures could be validated for use in a secure environment, because commercial off-the-shelf components could not be considered trusted. However, Rushby & Randell [11] and Stoneburner & Snow [12] have shown that untrusted, single-level systems can be incorporated into a multi-level distributed system (such as may be found in a real-time computer system), and the result can still be validated. Thus, an open system architecture is a viable alternative to wholly proprietary systems even in a secure computing environment.

A high-level block diagram of an avionics system is shown in Figure 1. An expanded view of the integrated processor is shown in Figure 2a, while Figure 2b depicts one possible implementation of one of the integrated processor's signal processors. To reduce the size and weight of the avionics system, to reduce the number of unique processor designs, and to provide increased fault tolerance, the data processors and signal processors are interchangeable and assigned to tasks dynamically.

In Figure 1, assume that one of the data links operates at a lower security level than the RADAR subsystem. Both may utilize different data processors or signal processors (see Figure 2) at the same

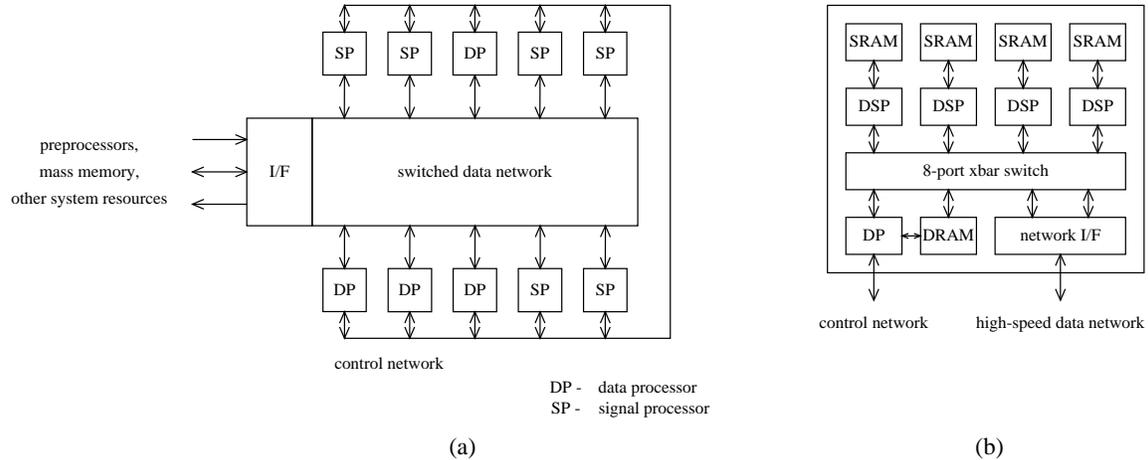


Figure 2: Block diagram of (a) an integrated processor and (b) a signal processor.

time, passing information across the data and control networks at different security levels. They may even use the same data processors or signal processors at different times. The data processors and signal processors must be capable of operating at various security levels and there must be safeguards built into the hardware and software that prohibit access to secure data by tasks that are not cleared for access to that data, whether that data exists in network messages or in processor memory as the result of a previous task. Potentially, the four digital signal processors (DSPs) shown in Figure 2b operate at different security levels simultaneously.

Important Differences

Traditional systems contain users and user-created objects, including files and processes running on behalf of the user. In contrast, the core processing elements of real-time embedded systems typically consist of a static set of well-defined processes and processing elements. Without such a well-defined set it is hard to ensure the required real-time performance constraints of the system. Similarly, it is unlikely that the users of the system (if any) will dynamically create new processes or objects (such as files). Although real-time systems may involve dynamic execution, resource allocation and scheduling, the possible behaviors of the system are constrained to a well-defined set of processes with specific limits on numbers and classes of each created object. A properly maintained embedded system will not have users downloading software from the Internet to execute on the system, or creating new data files with discretionary access controls. Instead, users can often be viewed as direct extensions of the computer system or considered to be separate subsystems executing at a single authorization level. Their interface to the system is through a specific physical interface, accessed through physical authentication controls (a pilot may not supply a password to start the aircraft, but needs to pass an armed guard instead).

Processes should be part of a well-defined static set rather than created dynamically (i.e., not installed or newly compiled by users). Although the run-time behavior of these processes and associated resource allocations may be input driven, normally the relationship between processes and associated protection levels is predetermined and fixed. Thus, information flow and protection controls between processes can be statically defined. This eliminates the need for a discretionary access control policy and permits all access control to be managed by a mandatory policy.

The environment in which a given system operates affects the guidelines which are pertinent to its

evaluation. We make the following assumptions about this environment: (1) Every individual in a position to observe human-readable output (such as that provided by gauges and data records) will be cleared at the appropriate level. (2) The system will not be vulnerable to physical tampering (although may be vulnerable to hardware damage or failure).

TCSEC Guidelines that should be modified

Because of the limited nature of the real-time embedded systems we are examining, some of the TCSEC guidelines are irrelevant or less important.

Discretionary access control We believe that traditional discretionary access control will not be a particularly useful tool in many real-time embedded systems for several reasons:

- Subjects in traditional TCSEC systems generally consist of human users, computers or other mechanical systems, and processes. In a real-time embedded system, we have well-defined processes with well-defined roles. Thus, we expect that if we were to associate sensitivity labels these processes such labels would never change and would be known in advance
- Traditional TCSEC systems expect objects to be created dynamically, and the number and ownership/sensitivity of these objects is not expected to be known in advance. This is one of the reasons that discretionary access controls are a useful tool. However, in a real-time embedded system, we can assume that all objects that will ever exist are present or planned at its inception. In addition, the role that each object plays in the system will be known in advance. The major justification for such assumptions lies in the need for precise timing between real-time system components. In order to assure properly timed interactions between components, it will be necessary to know all types of interactions in advance.

Because of the prior knowledge outlined above, we believe that mandatory access controls will be sufficient (and preferable) for defining information flow permissions between subjects and objects, that these relationships will not need to change dynamically, and hence discretionary access controls will be unnecessary.

Identification and Authentication Identification and Authentication will not be important within our real-time embedded systems. Since we are assuming the system is invulnerable to physical tampering, the only processes that exist will be those that are included by the developer. This does, of course, presuppose that, if we are using multi-vendor components, we can trust those components not to introduce intrusive processes. One caveat is that we may need identification and authentication *between* components of real-time systems in some situations. For example, if the avionics system is receiving targeting or mapping information from an external source, it will be important to have some assurance that the incoming information is trustworthy. However, there should not be any intruders in the usual sense.

TCSEC Guidelines that might require modification

Subject Sensitivity Labels If subjects are defined as human users, then subject sensitivity labels may not be necessary, since we are not expecting more than one category of subject to interact with the system during normal operation (the pilot). However, if it would be useful to have categories of subject (such as maintenance, pilot, co-pilot), then subject sensitivity labels will be useful. Also, if processes are considered subjects, these labels will be needed as well.

Trusted Path If we are assuming no human users and no physical breach of the system, then there ought not be any possible tampering of information between components. However, this may not be true if we have untrusted vendor components within the system itself.

Audit Audit should probably be included for purposes of maintenance and performance checks, and possibly for validating the actions of outside vendor components. However, if we do not expect to have distinguishable human users and do not expect intrusive processes, we will not need audit records that are aimed at identifying misuse from these sources. The real-time system must still maintain an audit trail of accesses to the objects it protects. Read access to the audit data is limited by physical means. The variety of events that must be recorded is reduced for a real-time embedded system because of the lack of dynamically created objects.

TCSEC Guidelines that should remain unchanged

Reuse The complex avionics systems now being developed usually rely on reuse of objects such as memory and processors for efficient operation. Thus, criteria providing guidelines for object reuse will still apply.

Labels, Label Integrity, Device Labels, Exportation Since mandatory access controls will probably still be needed, subject and object labels (and label maintenance) will still be needed. If our real-time system includes multi-vendor components (as seems likely), then it will be particularly important to continue to address the issue of labeling devices and passing labeled objects between devices.

Assurance, Documentation These categories of the guidelines are system independent. They provide a mechanism to ensure that reasonable effort was made throughout the system life-cycle to provide correct implementation and operation of the system with respect to the security policy and accountability guidelines. Although the amount of documentation, testing, and verification involved may vary between systems, these requirements must be met for all systems to be evaluated.

Conclusion and Ongoing Work

When compiling a review of TCSEC categories we found that some of the standard TCSEC guidelines were not applicable to a real-time embedded computer system. This discovery led to the review

Table 1: Evaluation Criteria Summary for Real-Time System

Criteria	Appropriate for Real-Time	Comments
Security Policy		
<i>Discretionary Access Control</i>	No	Substitute MAC
<i>Object Reuse</i>	Yes	Memory, processors shared.
<i>Labels, Label Integrity</i>	Yes	Needed for MAC.
<i>Exportation of Labeled Information</i>	Yes	Vendor-supplied components along common bus
<i>Exportation to Multi/Single-level Devices</i>	Yes	“”
<i>Labeling Human Readable Output</i>	Yes	Flight data recorders, printouts.
<i>Mandatory Access Control</i>	Yes	Predefined relationships.
<i>Subject Sensitivity Labels</i>	Probably Not	<i>Subjects</i> are defined as <i>human users</i>
<i>Device Labels</i>	Yes	Proprietary and vendor-supplied.
Accountability		
<i>Identification and Authentication</i>	None or Limited	Assumption 2; also, components untrusted, subjects cannot create processes, hence objects will be statically identifiable
<i>Audit</i>	Limited	Functionality/performance checks, covert channel detection.
<i>Trusted Path</i>	None or Limited	Unnecessary if no human users;
Assurance		
<i>System Architecture, Integrity</i>	Yes	
<i>Security Testing</i>	Yes	
<i>Design Specification and Verification</i>	Yes	
<i>Covert Channel Analysis</i>	Yes	
<i>Trusted Facility, Config Management</i>	Yes	
<i>Trusted Recovery</i>	Yes	
Documentation		
<i>Security Feature User's Guide</i>	Yes	
<i>Trusted Facility Manual</i>	Yes	
<i>Test, Design Documentation</i>	Yes	

of major TCSEC categories and their applicability to real-time embedded computer systems as presented in this paper. In summary this review points out that certain of the TCSEC guidelines, such as discretionary access control, user authentication, and export labels may be trivially satisfied or not even implemented in a real-time system. Although the claims in the review are generic, we are currently evaluating concrete examples to demonstrate the application of the TCSEC guidelines to specific instances of real-time systems. The work presented in this paper is just the first phase of a larger project that involves the analysis and interpretation of security guidelines for real-time embedded computer systems. We are working on the following related projects:

Formal Specification and Verification of Real-Time Systems. This project involves the use of formal specification and verification techniques for the security analysis of real-time systems. Currently we are investigating the analysis and specification of a real-time embedded avionics control system. The system involves a collection of processing units (potentially off-the-shelf) connected through a shared bus (as depicted in Figures 1 and 2). The paper [3] presents details of a high-level formal specification of the system, including information flow protection.

Formal Mapping of the TCSEC Guidelines to Real-Time Systems. We are currently involved in a project, which is a direct extension of this paper, to provide a mechanism for formally mapping the TCSEC guidelines to real-time systems. Although this project initially involves the avionics system discussed above, we plan to extend that work to other real-time control and manufacturing environments.

References

- [1] Department Of Defense Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria*, August 1983.
- [2] R. J. Fiertag, K. Levitt, and L. Robinson. Proving multilevel security of a system design. In *Proc. Symposium on Operating System Principles*, pages 57–95, 1977.
- [3] J. Alves-Foss, G. Saghi, D. Frincke and S. Ghantasala. Multilevel data security for real-time, embedded computer systems: A case study. In *Third AMAST Workshop on Real-Time Systems; Models, Properties and Control*, March 1996.
- [4] J.A. Goguen and J. Meseguer. Security policies and security models. In *Proc. IEEE Symposium on Security and Privacy*, pages 11–20, 1982.
- [5] J.A. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. IEEE Symposium on Security and Privacy*, pages 75–86, 1984.
- [6] J. McClean. Security models and information flow. In *Proc. IEEE Symposium on Security and Privacy*, pages 180–187, 1990.
- [7] J. McClean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. IEEE Symposium on Security and Privacy*, pages 79–93, 1994.
- [8] D. McCullough. Foundations of Ulysses: The theory of security. Technical Report RADC-TR-87-222, Odyssey Research Associates, Inc., July 1988.
- [9] D. McCullough. Noninterference and the composability of security properties. In *Proc. IEEE Symposium on Security and Privacy*, pages 177–187, 1988.
- [10] G. Norris. Boeing’s seventh wonder. *IEEE Spectrum*, 32(10):20–23, October 1995.
- [11] J. Rushby and B. Randell. A distributed secure system. *IEEE Computer*, 16(7):55–67, 1983.
- [12] G.R. Stoneburner and D.A. Snow. The Boeing MLS LAN: Headed towards an INFOSEC security solution. In *Proceedings of the 12th National Computer Security Conference*, pages 254–266, October 1989.