PAT Guidance Working Group

Form and Content of Vendor Design Documentation

NATIONAL
COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

May 1994

Report No. TBD
Library No. TBD

FOREWORD

The National Computer Security Center is issuing the *Form and Content of Vendor Design Documentation* and the *Form and Content of Vendor Test Documentation* to define process requirements and to provide guidance for meeting the C2-B1 design documentation and test documentation requirements, respectively, of the Department of Defense Trusted Computer System Evaluation Criteria (DoD 5200.28-STD). The guidance contained in the documents provides specific approaches for creating comprehensive views of the vendor's design and test documentation and establishes a set of deliverables needed to follow the approaches.

In addition to the current set of deliverables, the following documents are now required for systems entering the Trusted Product Evaluation Program: the Architecture Summary Document, the Interface Summary Document, and the Test Matrix Document. The *Form and Content of Vendor Design Documentation* establishes requirements for the Architecture Summary Document, the Interface Summary Document, and an expanded Philosophy of Protection. The *Form and Content of Vendor Test Documentation* establishes requirements for the Test Matrix Document.

The goals of the approaches described for vendor design documentation and test documentation are the following: to provide clearly defined, reliable methods for meeting the design and test documentation requirements of the Criteria; to allow the vendor to use as much of the existing system documentation as possible and to quickly identify incomplete or missing documentation; and to mitigate or eliminate significant delays in the evaluation schedule due to the lack of adequate design and test documentation. In an effort to provide specific guidance, recommendations are made concerning the detailed content of vendor design documentation and test documentation. Vendors who follow the guidance closely will more likely complete the evaluation in an efficient manner.

The approaches in the two documents were established by the Process Action Team (PAT) Guidance Working Group (PGWG) with assistance and significant comment from members of the Interpretations Working Group (IWG) and the Trusted Product Evaluation Program (TPEP) community. The efforts of the PGWG are part of a continuing initiative to improve the process and the overall quality of TPEP. Vendors are encouraged to provide feedback on the PGWG documents to evaluation teams or directly to the National Computer Security Center. As experience is gained in the use of the guidance and as sufficient feedback is obtained from the vendor and evaluator communities, the documents will be revised as necessary.

ACKNOWLEDGMENTS

Acknowledgements are given to the primary authors of and contributors to this document: Anne Lindell and Kraig Meyer of The Aerospace Corporation; Brett Borgeson, Robin Medlock, Ronald Watro, and Robert Williamson of The MITRE Corporation. In addition, the members of the Interpretations Working Group as well as other members of the Trusted Product Evaluation Program (TPEP) evaluation community have provided invaluable guidance and insightful comments on earlier revisions of this document.

TABLE OF CONTENTS

FOREWORD **iii**

ACKNOWLEDGMENTS **iv**

1 Introduction **1**
1.1 Approach to Design Documentation 1
1.2 The High-Level Design Documentation 1
1.3 The Detailed Design Documentation 3

2 The Architecture Summary Document **5**
2.1 System Overview 5
2.2 TCB Hardware Architecture 7
2.3 TCB Software Architecture 11

3 The TCB Interface Summary Document **25**
3.1 TCB Interface Overview 25
3.2 TCB Interface Summary Tables 30
3.3 Example 31

4 The Philosophy of Protection Document **33**
4.1 Subjects/Objects (TCB-Protected Resources) 33
4.2 Statement of Policy 34
4.3 TCB Protection Mechanisms 36
4.4 Assurances 39
4.5 Requirements 39

A Acronyms **43**

B Bibliography and References **45**

This page intentionally left blank

FIGURES

1.1 Overview of Design Documentation 2

2.1 ConvexOS/Secure System Block Diagram 8
2.2 Example Processor Summary Diagram 9
2.3 UNIX Example 14
2.4 UNIX Example (concluded) 15
2.5 Monolithic Example 16
2.6 Monolithic Example (concluded) 17

3.1 UNIX TCB Interface Structure 27
3.2 IBM VM/CMS TCB Interface Structure 27
3.3 Integrating X11 Windows as a TCB Interface 29

4.1 C2 I&A Requirement Example 40
4.2 C2 Object Reuse Requirement Example 40
4.3 B1 Labeling Human-Readable Output Requirement Example 41

TABLES

3.1 Format of Interface Tables 30
3.2 UNIX Process Management Example 31
3.3 DEC VMS I&A Interfaces 32

This page intentionally left blank

Chapter 1

Introduction

The Trusted Product Evaluation Program (TPEP) evaluation process for products in the lower assurance classes (C2-B1) separates design advice for security features from actual evaluation of a trusted product. The purpose of this report is to specify the form and content of the vendor design documentation that is required before an evaluation can begin. The requirements for vendor design documentation are based on the Trusted Computer System Evaluation Criteria (TCSEC) and accumulated evaluation experience.

The design documentation requirements detailed here are intended to make it easier for standard vendor design documentation to be used for evaluation purposes. The new approach requires summary documents (defined below) that provide a "roadmap" to the architecture, interfaces, and protection features of the product. The roadmap documentation allows the evaluation team to make the best use of the vendor's standard documentation. It also helps determine whether there are gaps in the existing documentation and aids in organizing the material for the version control required by the TPEP Rating Maintenance Phase (RAMP).

The specific information in this document concerns evaluation of operating systems at the C2 and B1 levels. Detailed instructions for the form and content of design documentation necessary to evaluate operating systems at higher assurance levels, networks, and database management systems will be developed later. The general information in this document should be useful for evaluations of all types.

1.1 Approach to Design Documentation

Trusted product design documentation is broken into two parts: *high-level design documentation* and *detailed design documentation* (see Figure 1.1). The form and content of the high-level design documentation is described in detail in this report. The high-level documents provide concise descriptions along with references that accurately indicate where more detailed information can be found. This high-level information not only fulfills the roadmap role described above, but it also provides critical support towards determining whether a product is ready for evaluation, and furnishes a starting point in the evaluation team's development of the Initial Product Assessment Report (IPAR).

1.2 The High-Level Design Documentation

The high-level design documentation is part of a new approach toward meeting the TPEP design documentation requirements. As shown in Figure 1.1 there are three required high-level design documents: the Architecture Summary Document (ASD), the Interface Summary Document (ISD), and the Philosophy of Protection (POP). The ASD, ISD and POP present three distinct views of the same system. These documents must be consistent with each other and they must all refer to each other to specify explicitly where

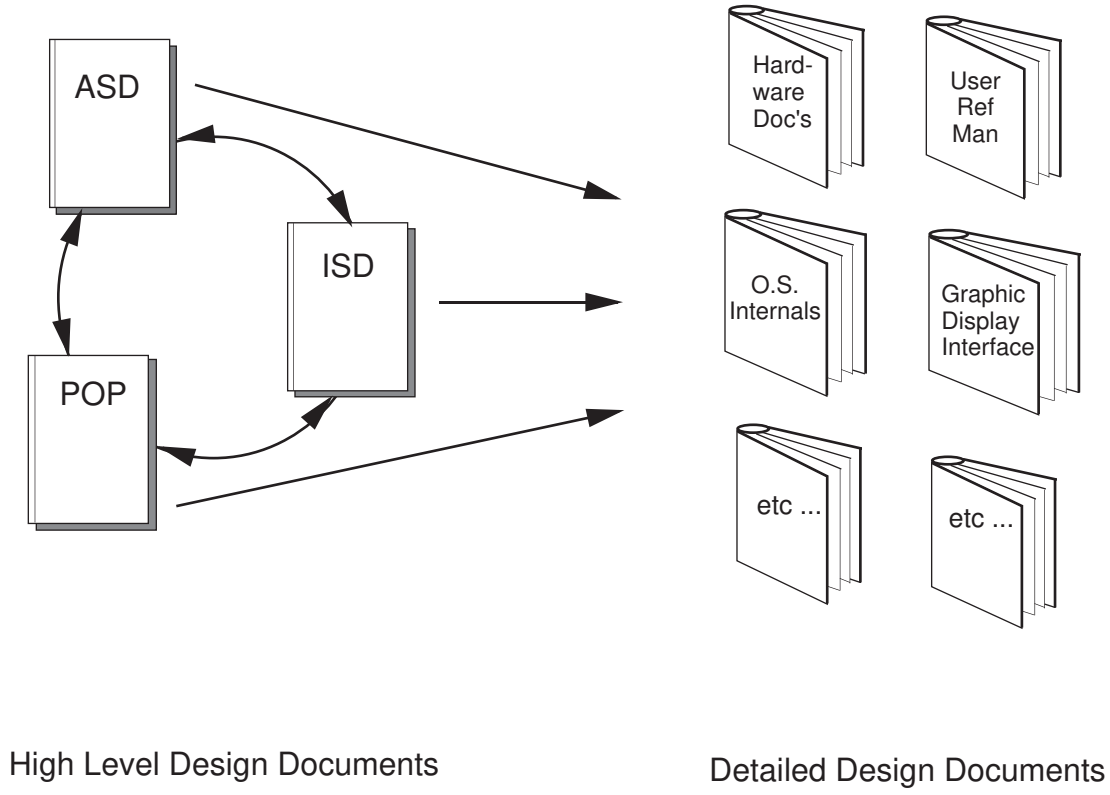

Figure 1.1. Overview of Design Documentation

the other viewpoints for the same system components can be found. The length of these documents will depend on the complexity of the candidate product, but they should normally comprise no more than a small fraction of the vendor's product documentation. The required content and recommended format of the three documents is described in this report.

The ASD provides an overview of the implementation of the product. The ASD describes the product's overall structure and the Trusted Computing Base (TCB) hardware and software. It identifies and describes the logical subsystems in the product and includes accurate references to further information. A detailed description of the ASD is given in chapter 2.

The ISD summarizes the services available to the users of the product. This is quite distinct from the ASD, which summarizes the logical subsystems of the product. The ISD is *not* the same as the product's reference manuals: it summarizes, and points to the reference manuals, but it does not describe interfaces in detail. A description of the ISD is given in chapter 3.

The POP is a revised version of a document that has always been part of TCSEC design documentation. The primary new feature of the POP is a step-by-step analysis of how the candidate product meets each of the TCSEC requirements. Chapter 4 describes the content of the POP.

1.3 The Detailed Design Documentation

The detailed design documentation comprises the bulk of the design material. It is produced and maintained by the product vendor and possibly other organizations (e.g., microprocessor manufacturers). The majority of this material will have been generated by the companies during the course of their ordinary product development. However, if detailed design documentation has not been maintained by the companies involved, then it must be written before the evaluation can begin.

The level of detail required in the detailed design documentation is difficult to define precisely. In the rainbow series document on design documentation, the following "rule of thumb" is given:

> ... the level of detail of design documentation should be sufficient to permit an individual with a degree in Computer Science, Electrical Engineering, or the equivalent with knowledge and skills in programming, hardware or firmware development to understand the system design and be able to design system modifications without adversely affecting trustworthiness of the system. [13]

Note that this rule refers to designing a change in the product, not implementing the change.

Examples of the level of detail typically needed in design documentation are: for hardware, logic flow diagrams, but not schematics; for software, a description of the data structures and associated algorithms involved with access checks[1].

The best general guidance concerning the level of detail required is that it will vary depending on the importance to security of the area under discussion. Non-TCB software needs to be discussed only to the extent necessary to establish the fact that its exclusion from the TCB is justified. The detail necessary to establish this depends on the vendor's rationale for identifying the TCB (contained in the ASD).

The level of documentation on TCB software can also vary. Parts of the TCB may play no role in enforcing security, and these portions require less detail than security-enforcing software. Some additional guidance on the level of detail required is provided in the next three chapters of this document. As these chapters define the high-level documents, they discuss the supporting detailed design information that needs to be available.

[1] All levels of detail including the implementation in code or schematics may be reviewed during evaluation if necessary to address security concerns.

This page intentionally left blank

Chapter 2

The Architecture Summary Document

The purpose of the Architecture Summary Document (ASD) is to summarize the design of a candidate product and to provide references to the vendor's detailed design documentation indicating where further information can be found. The topics covered in the ASD will be similar to the topics in the System Overview, Hardware Overview, and Software Overview sections of recent National Computer Security Center (NCSC) Final Evaluation Reports (FERs). (See [6, 8, 9, 11, 12] for recent examples.) Because the ASD is an overview together with pointers to references, it may actually contain less detail on some topics than a FER. Much of the information required in the ASD is the same as that required by the Trusted Product Evaluation Questionnaire [16], but the ASD is intended to provide more comprehensive information about the structure and mechanisms of the candidate product that support the evaluation requirements.

The information required in the ASD is described here in three sections:

1. The System Overview section, which summarizes the system as a whole and identifies those components that comprise the TCB
2. The Hardware Architecture section, which summarizes the hardware structure and design of the product and provides references to detailed design documentation for those features that support the evaluation requirements
3. The Software Architecture section, which summarizes the software structure and design of the product and provides references to detailed design documentation for those features that support the evaluation requirements.

The detailed design documentation that is referenced in the Hardware Architecture and Software Architecture sections of the ASD may describe all aspects of the candidate product, but it must provide details for those aspects that are security relevant (i.e., those features that support the Security Policy, Accountability, and System Architecture requirements of the TCSEC).

The sections below describe in more detail each of the proposed sections of the ASD.

2.1 System Overview

The System Overview section of the Architecture Summary Document presents an overview of the entire candidate product. It introduces the product's major logical subsystems and describes how those subsystems work together. It also identifies which logical subsystems are in the TCB and provides the rationale for TCB inclusion or exclusion.

The following subsections are recommended in the System Overview section of the ASD. They serve to introduce the candidate product from a functional point of view, instead of specifically from a security point of view.

2.1.1 History

This subsection summarizes the history of the candidate product line, both hardware and software. It identifies the major hardware product line transitions and describes when major hardware and software security features were added. If the candidate product includes another vendor's hardware or software, this subsection provides a summary of how that hardware or software has been incorporated into the candidate product and any modifications that have been made. It also identifies whether any previous versions of the hardware or software components have been evaluated by NSA.

2.1.2 High-Level Product Overview

This subsection identifies the type of system (e.g., Monolithic System, Layered System, Virtual Machine, Client-Server Model, Network Operating System, Compartmented Mode Workstation) and its intended environment and use. It describes how users interact with the system, what major resources are managed by the system, and what mechanisms are provided for users to communicate with each other, if applicable.

This subsection identifies the major pieces of the candidate product, breaks them down into logical subsystems, and summarizes the major functions of each logical subsystem. Each logical subsystem itself may be a product (e.g., a base operating system, a security manager, or a resource management subsystem), or it may be a set of logical functions within a single product (e.g., the kernel, the file system manager, the memory manager, or a trusted process). Note however that logical subsystems should not be created solely on the basis of productized components. For example, if Access Control Lists (ACL) are added to a file manager by the incorporation of an add-on security product, the logical subsystem should be the file system manager including the ACL modification.

2.1.3 TCB Identification

Typically, a logical subsystem is included in the TCB if it:

- Enforces the system security policy or contains data used in enforcing the security policy
- Possesses or may acquire software privilege that allows it to bypass the system security mechanisms
- Is capable of executing privileged hardware instructions
- Must function correctly to support the system security mechanisms
- Must be used by the System Administrator to administer the system in a secure manner.

This subsection identifies which logical subsystems of the candidate product are part of the TCB. These TCB logical subsystems, in turn, provide the basis for the structure of the Hardware Overview and Software Overview sections of the ASD, and also provide the basis for the structure of the ISD. For each logical subsystem, a rationale must be provided for its inclusion or exclusion from the TCB.

This section must specifically address software library routines or system service routines that are included in the candidate product and state whether they are included or excluded from the TCB.

2.2 TCB Hardware Architecture

The TCB Hardware Architecture section of the Architecture Summary Document summarizes all features of the hardware, identifies those that are security relevant and why, and provides references to detailed design documentation for the security-relevant features of the hardware. This detailed design documentation must provide enough information to determine how the hardware components support the TCB software to meet the TCSEC requirements. Typically, the hardware provides critical support to meet the TCSEC System Architecture requirements, but it also supports the TCB's Security Policy and Accountability requirements. Throughout this section, each feature must be identified as used or not used by the TCB software. If a feature is used by the software, how the feature is used is described in the Software Architecture section.

2.2.1 Hardware Overview

The Hardware Overview subsection provides a high-level description of all hardware in the candidate product, including a high-level block diagram of the hardware components. This diagram must depict all processing units (including CPUs, math coprocessors, etc.), memory controllers, memory, I/O processors, I/O controllers, I/O devices (e.g., printers, displays, tapes, communications lines), and the communication mechanisms between them. A brief description of each component's function must be provided. If the hardware can be configured in more than one way, these different configurations must be described.

Figure 2.1 shows a portion of a high-level block diagram taken from the ConvexOS/Secure Final Evaluation Report [4]. (The I/O devices are not shown in the figure.) The content of the FER describes in more detail each of the subsystems depicted in the diagram.

2.2.2 Processor Hardware Architecture

The Hardware Architecture subsection provides a description of the processor architectures. If more than one model of a hardware family is to be included in the candidate product, each model must be described and differences between models must be summarized. If there is a common architecture for all models of a hardware family, this also must be described, with a reference to where the commonality is defined and an explanation of how architectural compliance is determined.

The following sections describe the type of information that must be provided for each hardware family, if applicable. Not all sections described below may apply to all systems. Whenever a topic is applicable, however, the relevant information must be provided.

Processor Summary Diagram

The ASD must provide a general description of each CPU utilized by the candidate product. The description should include or make reference to a block diagram that indicates the relationship among each element of the CPU (e.g., the Instruction Processor, Microcode Processor, Memory, Cache, Memory Mapping, or Address Translation Unit). Figure 2.2 shows an example of a processor summary diagram, based on the Intel MC68020 [15].

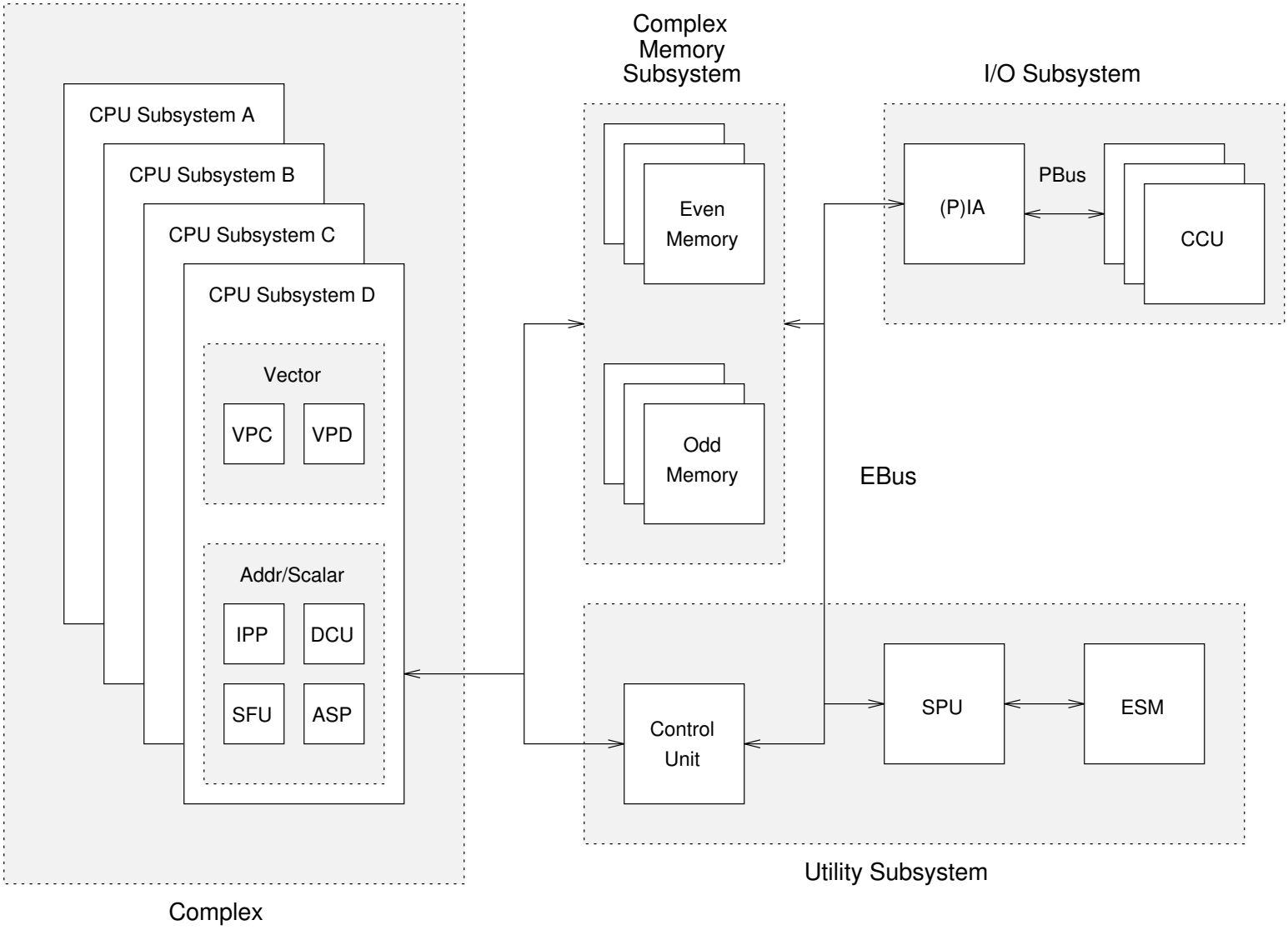

Figure 2.1. ConvexOS/Secure System Block Diagram

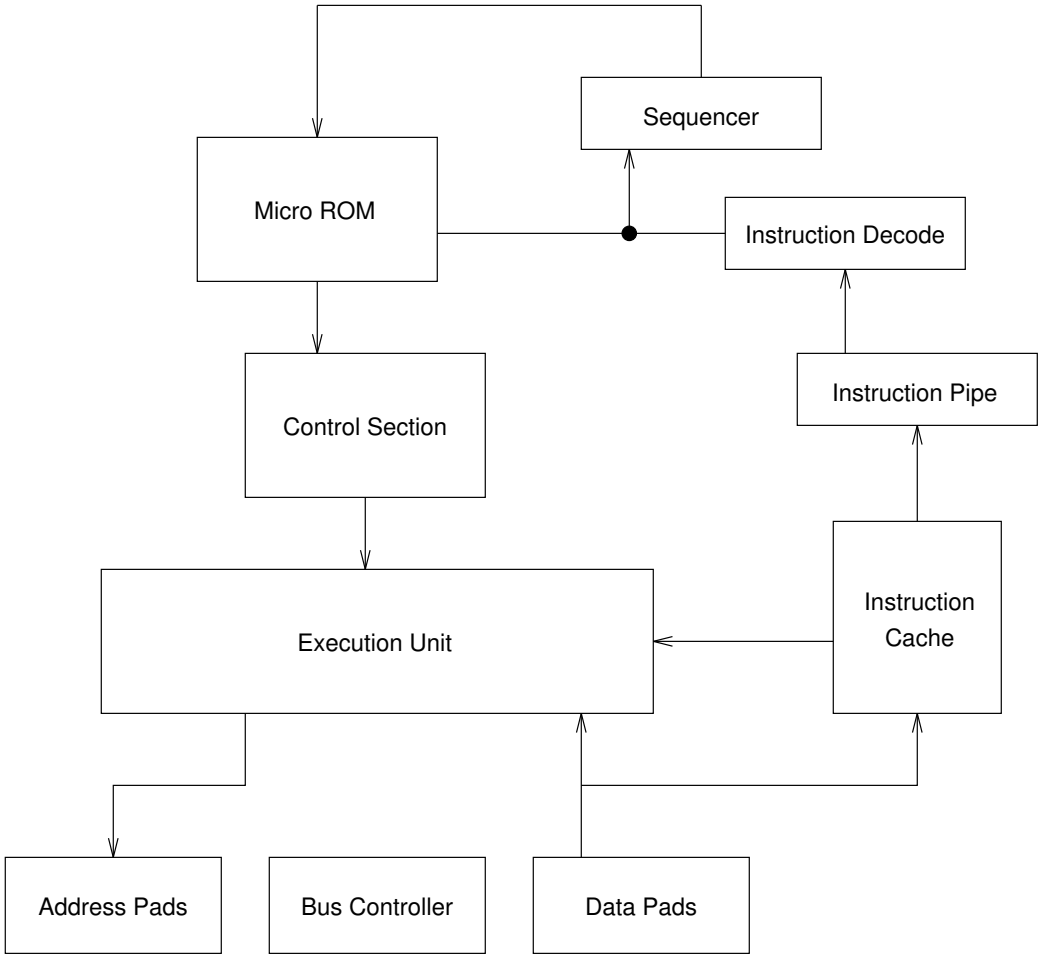

Figure 2.2. Example Processor Summary Diagram

Firmware or Microcode

If the hardware configuration contains firmware or microcode, the ASD must summarize its functionality. If the firmware or microcode is loadable, references must be provided to detailed design documentation that describes how it is loaded and protected from modification.

Hardware Protection Mechanisms

The ASD must provide a summary of the hardware features that are used to support TCB isolation and process separation requirements, along with references to detailed design documentation that describes the hardware privilege mechanism (e.g., user/kernel mode, multiple privilege levels, or problem/supervisor state). The ASD must state which domain(s) of execution are used by the TCB and which domain(s) are used by user processes. If not all hardware domains are used by the candidate product's software, or are not available to user processes, these must be identified, and an explanation provided of how these domains are protected from being used. The ASD must also describe the steps that occur during a transition between domains and establish how the flow of control occurs (e.g., a high-level description of domain transitions).

Instructions and Registers

The ASD must contain a brief description of the instruction streams and registers provided by the hardware. References should be given to the instruction interface summary in the ISD and to more detailed design documentation that fully defines the hardware interfaces. If two or more different hardware families are to be evaluated, this subsection describes each hardware family separately.

The detailed design documentation must describe the instruction set of the processor or processor family, including a list and a brief description of each privileged instruction. For each hardware instruction, it must state the privileges required to execute the instruction. Examples of privileges include the machine state, the executing ring/segment/domain/privilege level, physical memory location of the instruction, etc. If the configuration contains multiple processors, processor coordination must be described.

The detailed design documentation must also describe program-visible registers. This includes the registers that an untrusted user can manipulate and those that are manipulatable only by the TCB software. It must describe any privilege required to manipulate these registers. Furthermore, if particular registers are used to support TCSEC requirements, their use must be described. Examples of this sort include the program status registers, control registers, and descriptor registers.

Address Translation

The ASD must summarize how address translation is accomplished by the hardware (i.e., logical/virtual to physical) and reference detailed design documentation that describes the translation process, including data structures for supporting segmentation and paging (e.g., segment definition tables, page table structures, page protection attributes).

Interrupt Handling

The ASD must summarize and reference detailed design documentation that describes hardware support for handling interrupts and exceptions. This detailed design documentation must list all interrupts and traps, and describe how they occur and how they are meant to be serviced by the system. The documentation must identify the mode in which interrupts are handled, how transition between modes is managed if an interrupt occurs, and how interrupt data structures are used.

I/O Hardware

The ASD must provide a block diagram of the I/O hardware architecture and reference detailed design documentation that describes any I/O buses, controllers, and devices in the candidate product configuration (e.g., whether they are directly accessible by users, whether they contain user-programmable features, or whether they may retain any user information after the information has been processed). The detailed design documentation must describe processor support for I/O and the mechanisms (if any) that are used to prevent users from directly executing I/O instructions. It also must describe how I/O is initiated.

If the candidate product contains Direct Memory Access (DMA) controllers or devices, the ASD must reference documentation that describes how address translation is done for the DMA controllers or devices. If the translation is done in hardware, references to detailed design documentation should be supplied. The referenced information must specify whether the address translation is done through the memory address translation unit or whether the logic is part of the controller, and how the address translation maps or tables are initialized. If the translation is done in software, references should be given to the appropriate locations in the Software Architecture section in the ASD.

If the I/O hardware includes any embedded processors, the referenced detailed design documentation must include descriptions of the instructions and registers that are provided, any loadable firmware or microcode, the hardware protection mechanisms, how address translation is accomplished by the hardware, and how interrupts are handled.

System Console or Control Hardware

The ASD must summarize and reference detailed design documentation that describes any hardware components that are used for devices to which only trusted users have access. This description must identify what the device does and how its use does not compromise the TCB (e.g., may run in a maintenance mode, may be accessed through privileged I/O instructions, may be physically protected). If the candidate product includes workstations, personal computers, or terminals—either embedded in the system or as an add-on—(e.g., consoles), they and their operating system must be identified and described, and a justification must be given for why they don't compromise the TCB.

2.3 TCB Software Architecture

The TCB Software Architecture section of the Architecture Summary Document summarizes the TCB software, identifies those portions of the software that are security relevant and why, and provides pointers

to detailed design documentation for the security-relevant features of the software. The detailed design documentation must be sufficient to support the argument that the security policy, accountability, and relevant assurance requirements of the TCSEC are met. The detailed design documentation should describe how the structure of the TCB guarantees that objects under the TCB's control can be accessed only via requests made through the TCB, that the access mediation is uniform and consistent with the system policy, and that mediation is performed for each class of objects available at the TCB interface.

The recommended organization of the TCB Software Architecture section is into two parts: a TCB Software Overview and a Logical Architectural Subsystem Descriptions Section. The content of these two parts are defined by the following material. Section 2.3.1, TCB Software Overview, discusses, in general terms, the content of the TCB Software Overview. Section 2.3.2, Examples, provides two examples of TCB Software Overviews, one for a UNIX-based system and one for a monolithic operating system. Section 2.3.3, Logical Architectural Subsystem Descriptions, describes in detail the topics that must be covered in the Logical Architectural Subsystem Descriptions.

It is important to note that not all sections discussed below will apply to all candidate products. However, whenever a topic is applicable, the relevant information must be provided. In addition, there may be TCB-oriented topics that are not listed below but are applicable to a candidate product. Any such topics that a product vendor can identify should be discussed in the ASD.

2.3.1 TCB Software Overview

This section of the ASD provides an overview of the TCB software. It includes a description of the TCB structure, identification of privileged and unprivileged (with respect to hardware) software, and identification of TCB databases. This material provides the foundation for the descriptions needed in the Logical Architectural Subsystem Descriptions section of the ASD.

TCB Software Structure

This section of the ASD describes the structure of the TCB software. Usually, the structure of the TCB software is driven by the design type of the operating system (e.g., kernelized, layered, monolithic).

Software That Executes with Hardware Privilege

This section of the ASD identifies the more privileged hardware states (e.g., kernel, lowest layer, ring 0) and the TCB software that executes in each of these states. The roles played by this software within the TCB are summarized. Software that implements memory management, process management, interrupt handling, Interprocess Communication (IPC), and file system management usually falls into this category of TCB software.

Software That Executes without Hardware Privilege

This section of the ASD identifies the TCB software that executes in the least-privileged (or unprivileged) hardware state. This category of software includes software that executes with and without software priv-

ileges. Privileged (with respect to software privileges) software can be divided into software that executes with privilege regardless of the subject that executes it, software that functions differently when it is executed by a subject that possesses privileges versus a subject that does not possess privileges, and software that requires that the subject have privileges in order to execute it.

TCB Databases

This section identifies all the important security-relevant databases (e.g., user profile database, password database) and provides a brief description of their purpose.

2.3.2 Examples

Two examples of TCB Software Overview sections are given. The first example (Figures 2.3 and 2.4) considers a typical UNIX-based system. The second example (Figures 2.5 and 2.6) considers a monolithic operating system (UNISYS OS 1100 [10]). In the examples, roman text indicates actual wording that would appear in the TCB Software Overview. Italicized text is either a description of information that should be present in a particular place or instructions that assist in the writing of this section of the ASD.

2.3.3 Logical Architectural Subsystem Descriptions

The subsections below are an outline of topics to be discussed in the proposed Logical Architectural Subsystem Descriptions section of the ASD. The organization of these topics is established in the TCB Software Overview section. The topics below are organized based on a kernelized architecture; hence, there is likely to be a one-to-one mapping from the topics to the logical architectural subsystems. Note also that these logical architectural subsystems may be related to RAMP configuration items.

Memory Management

In this section, provide a functional description of the memory management architecture. Summarize the security-relevant aspects of the memory management subsystem, focusing on the major data structures used to implement the virtual address space. Portions of this discussion provide the basis for the process isolation argument. Explain whether memory can be shared and what assures address space isolation when memory is not being shared. Include an explanation of the mechanism that prevents untrusted users from accessing kernel memory. Present an overview of the memory addressing modes and issues involving virtual address translation. Provide a discussion of the typical address space layout for processes (i.e., tasks or threads of execution), describing the process components and the memory management structures used to maintain the components. Lastly, describe the candidate product-wide address space.

Memory System Overview. Identify the modes of addressing that are available (e.g., absolute, real, virtual), and provide pointers to the detailed descriptions of the addressing modes, including how they differ from one another, what mode is used by the various system processes, and which modes are available to untrusted users. Identify and briefly describe the fundamental memory resource available in the virtual address scheme (e.g., page) and the physical addressable unit of memory to which it corresponds. Identify

---

TCB Software Overview

The TCB is organized into two major functional units separated by a well-defined interface:

- The body of code that executes in the hardware mode with the most privilege — the kernel.
- The body of code that executes in execution domains running without the hardware's privileged mode — the non-kernel TCB.

The system call interface separates these two components. This interface is the set of system services available to all processes that are invoked by the calling sequence of operating system traps. A set of C language library routines is provided that present a high-level language interface to the machine language instructions. These instructions implement the hardware trap that passes control to the privileged mode code that requests the appropriate service. The operating system itself consists of a single execution image loaded into the system's main memory at boot time.

System Kernel *In this section, summarize the role played by the system kernel within the TCB (i.e., in general, it performs resource management of system-wide resources).* All operations involving processes are controlled by a portion of the operating system called the system kernel (also known as the nucleus or core). The system kernel runs in the most privileged mode (i.e., privilege level 0). The kernel is organized into logical subsystems, layered as follows:

- At the top layer (the OS services or system call layer), these subsystems provide implementation of all the system calls.
- At the bottom layer (the machine-dependent layer), these subsystems handle the peculiarities of the hardware environment on which the operating system is running.

In between these layers are logical subsystems that provide abstractions of various software objects that are implemented using the resources of the machine-dependent layer.

Continued on the next page.

Figure 2.3. UNIX Example

The kernel is divided functionally into a number of logical subsystems. Each subsystem implements a set of software objects accessible to processes through the system call interface or to other kernel subsystems through internal kernel interfaces. The code, data structures, and interface definitions for each subsystem define the module structure of the kernel. The relationship between these modules is defined by the way functional subsystems interact and use the interfaces of the other subsystems to implement their services. *Identify each subsystem that is wholly or partially implemented in the system kernel, defining the major objects implemented, the interfaces into the subsystem, and the services of other subsystems used.* The subsystems include: Memory Management, File Management, I/O Management, Process Management, Device Drivers, Interprocess Communication, Audit, and Access Mediation Management.

Non-Kernel TCB Software The software that runs without hardware privilege is organized into processes. A process is an active entity such as a task or single thread of execution. Processes that implement some aspect of the system's security policy comprise the non-kernel TCB. A trusted process is a process that has been given a privilege that permits it to bypass system security mechanisms or manipulate security-critical data. *In this section, describe the role played by trusted processes in the TCB and the general environment.* The trusted programs included in the non-kernel TCB are organized into logical subsystems, each of which provides some service to other (non-trusted) processes. Each subsystem may include programs that are invoked automatically by the system and programs that users invoke to perform services. Trusted programs are normally distinguished by non-empty privilege sets, indicating that the process executed from that program has access to kernel services reserved for trusted applications. *Identify all trusted processes, both those that implement specific security mechanisms, such as a print spooler, and those that are trusted simply by inclusion in the TCB, such as an editor designated for administrative use.* The non-kernel TCB subsystems include: Authentication, File System Maintenance, System Backup, Mail, Batch Processing, System Administrator Functions, Printer Services, Terminal Management, Data Import/Export Management, Audit Data Management, and Security Policy Support Programs.

Figure 2.4. UNIX Example (concluded)

TCB Software Overview

The TCB is composed of the Executive that runs in processor ring 0, and 12 software modules that run in processor ring 2. *Identify the Executive and the other 12 modules in a block diagram with communication links between each.*

The Executive The Executive is the nucleus of the operating system and performs the following functions for the rest of the TCB as well as for general users:

- Central control — provides control of the hardware-oriented functions, including initialization, interrupt processing, I/O, and clock/timer support
- Processor management
- Main storage management
- Job scheduling
- Unit record device control
- Transaction processing
- Auditing
- Tape and file management
- Security - access control
- Support for automated recovery.

The Executive is structured into a set of logical subsystems. These subsystems are composed of independently compiled/assembled entities called elements. All of these elements are linked into a single executable entity that is loaded at initial program load. *Identify all Executive logical subsystems and describe their function as in the following:* The central security complex is a logical subsystem of the Executive. This complex contains the security validation routines and provides a unique interface for security validation by programs outside the TCB.

Continued on the next page.

Figure 2.5. Monolithic Example

The Executive provides services through the following three types of well-defined interfaces:

1. Executive Requests (ERs), an assembly language interface to the Executive itself, or to the security complex from within other parts of the Executive.
2. Executive Control Language (ECL), a command interface with the command interpreter inside the Executive. After reading a command, the interpreter passes control to the Executive component providing the requested service.
3. Operator-console keyins.

Each interface is identified and described in the ISD.

Software Privileges The TCB contains 12 logical subsystems that run in ring 2 (same ring as an unprivileged user), and these subsystems have no inherent access rights to objects different from the rights of ordinary users. However, these subsystems are given software privileges to bypass the security policy as needed to perform their functions. Software privileges, not to be confused with Processor Privilege (ring structure), fall into one of two categories, secured ERs and Privileges. A secured ER is a means to control access to a complete security-relevant TCB interface, whereas a Privilege is a mechanism to control access to a subfunction of a security-relevant ER.
In this section, provide a discussion of the two types of software privileges, to include: a reference to or a description of how the software privilege mechanism works, a list of each software privilege, the specific purpose for each privilege, and in which TCB component each privilege is used.

Non-Executive Trusted Software *In this section, identify each of the TCB logical subsystems that run in the unprivileged hardware state, describe or reference a description of its function in the TCB, and discuss why each logical subsystem requires privilege to perform correctly in the TCB. One such example follows:*

The File Administration System (FAS) provides backup, archival, tape management, and reporting functions for the file system. FAS operates under the privileges of the invoking user and extensive privilege is required to perform system-wide administrative actions.

Figure 2.6. Monolithic Example (concluded)

and provide pointers to a detailed description of the data structure used to maintain the real address units of memory. Identify the memory management data structures provided by the hardware (e.g., page table, segment table) and provide pointers to the detailed design documentation that describes how these structures are used by the software to manage addressable space, including virtual address to real address translation. Of the hardware memory management data structures, if there are any portions that are strictly software-specific management information (e.g., certain bits in a page table entry), identify them and provide pointers to the detailed description of how they are used. Also provide pointers to the detailed description of how the memory management data structures are used to map high-level abstractions, such as user files, executable program files, and shared libraries, into memory. If there is such a concept as shared memory (i.e., a mechanism by which processes can communicate directly with each other by sharing parts of their virtual address space), discuss the relationship to memory management. If it is possible to maintain multiple references to a common object until the object is modified (e.g., copy-on-write), point to the detailed design documentation that explains how this is implemented.

Process Address Space. Identify the data structures that are used for process memory management, noting which data structures make up the virtual address space of a process and how the data structures are addressed using the hardware memory management data structures. If the process address space can be organized into shared and private regions, describe the manner in which this is accomplished, noting the data structures used and the manner in which these data structures interact. If the regions can be categorized (e.g., as in UNIX, the text region, data region, stack region), describe these various categories of regions, including the purpose of the region, what hardware privilege modes (e.g., kernel mode, user mode, or a given ring) the region can be accessed in, whether the region is static in size, whether the region is write protected and how, and whether the region is shareable amongst processes. Note any restriction on the number of regions that can be associated with a process.

System-Wide Address Space. Describe how the system-wide virtual address space is related to individual process address spaces. Discuss how the hardware-defined spaces are used on a system-wide basis.

Process Management

Identify and briefly describe all structures that define a process. Provide pointers to detailed design documentation that discusses the security-relevant process data structures that are not addressable by a user process (i.e., they are managed by another entity). This includes a description of the data structure, along with a list of the fields of the data structure, used by the kernel to access information about an active process. It also includes a description of the data structure that contains environmental and I/O data about a process, as well as registers that should be saved during context switches and interrupt handling, and housekeeping data. In particular, describe how any process attributes (e.g., identifiers, clearance, current sensitivity label) that are stored in multiple instances are kept consistent.

In addition, define the context of a process. This usually includes the process's text, data, stack, and shared memory, the general purpose registers, control registers, process status word, and system structures that a process uses during its lifetime. Also describe when and how process context switching occurs.

Summarize and provide pointers to detailed design documentation that describes how processes are created and destroyed, including how process attributes are assigned at process creation time (e.g., in UNIX, some process attributes are inherited during process creation), how programs are invoked, how privileges

and/or special user identifier status are acquired by a process, how command language programs are invoked (and whether they can be privileged), and the manner in which a process can be notified of some type of asynchronous, software-related event.

Describe what process tracing does. Explain why it does not affect correct operation of the TCB if one attempts to trace a process with privilege.

Interrupt Handling

In this section of the ASD, describe how traps or interrupts (i.e., anything that can change the flow of processing) are handled by the software. Describe how interrupts are set, including what system structure is used to indicate that an interrupt has been set and what restrictions are placed on setting an interrupt. Also, describe how interrupts are generated and what can generate them (e.g., devices, controllers, other processors). Provide an overview of the structure of interrupts (e.g., interrupt levels 1-15 are reserved for use by system software, while levels 16-20 are reserved for use by devices). Provide pointers to the detailed design documentation that describes how interrupts are processed, including any data structures that are used in this processing.

File Management

The issues discussed in the following text apply in general to objects managed by the system. The file system was chosen to illustrate these issues since files and file-related entities are the objects that are most commonly found in evaluated systems.

A file is a named collection of data. The file system is that portion of the overall storage management system primarily responsible for managing files on secondary storage. It is responsible for providing, through possibly different mechanisms, the means for private storage of information and for sharing of information among users in a controlled manner. The file system may also provide and control access to other types of file system objects (FSOs) such as IPC objects and/or devices. The file system may provide services to these FSOs, such as buffering of data to the storage media and control of file system resources. In this section of the ASD, identify all FSOs included in the candidate product and how they are organized by the file system. Briefly describe how the file system is organized on secondary storage. Also, briefly describe how the file system is treated as a whole entity, including the interfaces available for manipulating the file system as a whole.

Provide pointers to the detailed design documentation where each FSO and its function in the file system is described. The detailed discussion provides the following information:

1. A description of how the FSO is accessed, including all data structures that are involved (e.g., per-process file table, system file table), and the security attributes used in making the security decisions when performing file system operations on the object

2. A description of how the secondary storage file system data structures map to the file system data structures in main memory, and how common information is kept consistent

3. A description (or a reference to such a description in the ISD) of the interfaces provided to manipulate the FSO (e.g., for files, there are interfaces to create, open, read, write, close, and delete the file)

4. A description of how FSOs can be deleted and continue to exist as objects if held open by some process, even though they can no longer be named, if applicable

5. A description of the mechanism that allows application of shared or exclusive locks to files or portions of files, if applicable.

I/O Management

In the ASD, briefly discuss the high-level implementation of system I/O, from the point of I/O initiation to the point where physical (low-level) I/O occurs. This discussion covers all entities to which the system performs I/O, which at a high-level includes objects such as files and devices.

Provide pointers to the detailed description of the policy for updating system data structures stored on secondary storage. Provide pointers to the detailed design documentation that discusses the mechanisms provided by the system for interfacing with devices, including character devices for I/O operations and block devices for file system I/O. This discussion should note which devices are reserved to the TCB or are managed by the TCB and which are directly usable. It should also describe any mechanism that provides exclusive access to a device.

Also, provide pointers to the detailed design documentation that discusses the low-level I/O implementation for local devices, showing how the device driver interface works. Not much detail is needed here, since all low-level I/O takes place only after access mediation has happened. The key point is that it is all under control of the TCB. As an exception to this general rule, if a device driver enforces a security requirement (e.g., process isolation), then detailed design information sufficient to verify correct enforcement of the requirement is necessary.

Inter-Process Communication

IPC occurs when one process transfers data to another process or when two processes are able to share data. In this section of the ASD, introduce all IPC mechanisms (e.g., message queues, semaphores, interprocess signalling, process tracing, process debugging) provided by the candidate product. For each IPC mechanism, describe briefly what it does, including whether and how the mechanism provides information to the receiver about the sender. Explain how the IPC objects are created and destroyed. Indicate the security attributes associated with the IPC objects. Identify and provide pointers to the detailed description of the interfaces (e.g., remove, status, set ACL) provided to manipulate these objects and their attributes. Provide pointers to the detailed design documentation that discusses the implementation of the IPC objects including whether they are persistent objects, and thus do not depend on processes holding them "attached."

Identification and Authentication

In this section of the ASD, summarize the significant Identification and Authentication (I&A) mechanisms, and describe these and other I&A features in terms of user login and the data required to maintain a user's identity for accessing the system and for creating a login user session. Describe the login sequence from the user's first action (e.g., typing a trusted path sequence or entering a user name) through the startup of the user's command processor (i.e., the shell — the system mechanism for communication between users and the system) and login session. Describe how the beginning and end of user sessions are audited. Describe how

the login terminal is made accessible to the user (and how access by other processes is revoked/prevented), and, for B1 products, how it is labeled. Identify all checks performed by the login process. These checks include, for example, that the user's password is valid, that the user's sensitivity label is within the user's clearance and terminal range, that time-of-day restrictions are not in force for the user's account, and that terminal access restrictions are not in force.

Describe how the login process establishes the initial set of mandatory, discretionary, and other security attributes to be associated with the user's session. Identify the set of security policy attributes assigned to the command processor. Generally, the command processor is started as the last step of the login process. Describe any significant actions taken by the login process after the user is authenticated, but before the command interpreter is started. Identify the information that is recorded about a user's session after login is successful and the user session is created. Describe how login failures are handled, including what information is recorded for each failed attempt and where it is stored.

Describe the user session termination process, including what information is recorded for the termination and where it is stored.

Identify and briefly describe the databases used in validating a user upon login, including the role these databases play in the login process and the significant information they contain.

Describe how user passwords are changed and how other I&A data may be changed by users.

Describe any mechanism that allows a user to start a process or command processor with a different user identity. In this description, identify the identification and authentication checks that are performed and any other restrictions, if any, placed on such an operation.

Batch Processing

In this section of the ASD, identify and provide pointers to the detailed design documentation that describes the mechanisms provided to perform batch processing (or any type of delayed processing), including any restrictions that are enforced both when submitting the job and when the job is executed. In the ASD, briefly describe how the attributes of the desired batch session are recorded so that the batch job is started with appropriate attributes (e.g., user identifier, sensitivity label privileges, audit control). In addition, provide pointers to the detailed design documentation that describes the batch process queue manipulation mechanisms and how batch processing is implemented.

Import and Export

In this section, identify mechanisms provided for the import and export of data (e.g., printer services, system backup and restore software, and tape archiving software). For each mechanism, identify any restrictions that are enforced when an attempt is made to use these mechanisms and any privileges that allow a user to perform additional or different import/export functions (e.g., backup and restoration of system data). Identify and provide pointers to the detailed description of any data structures used in determining whether an import/export device can be accessed, including how the data structures are used in the access checks that are made. Briefly describe the manner in which attributes of exported data (e.g., user identifier, ACL, sensitivity label) are maintained on the export medium and are properly restored upon import of that data. In the case of printers in a B1 configuration, briefly discuss the mechanisms used to prevent the spoofing of banner page labels and the mechanisms that ensure correct page labeling. Provide pointers to

the detailed design documentation that describes the implementation of each import/export mechanism from the initiation of an import/export request to the completion of the import/export operation.

Administrative Functions

In this section, describe services provided to update administrative and security databases and the mechanism employed to verify that only trusted administrators can use these services. If these services can be grouped by the general function they provide (e.g., printer subsystem interfaces, authentication subsystem interfaces), provide one subsection per readily identified grouping of these administrative services.

Auditing

In this section of the ASD, describe the services provided to initialize and control the audit subsystem, noting any restrictions that are enforced when using these services. Describe any utilities provided to administer the audit subsystem (e.g., audit data recovery tools, post-generation audit data analysis tools). Indicate any restrictions that are enforced when using these utilities.

Describe how the audit trail is stored, including the format (e.g., the audit trail consists of a header followed by audit records) and contents of the audit trail. Describe how the audit trail is protected and managed. Describe or provide a reference for the format and content of all possible audit records. Identify all possible auditable events. Also describe the mechanisms in place to prevent loss of audit data in the event that space for storing the audit trail fills up.

Provide pointers to the detailed design documentation that describes the functional characteristics of the audit subsystem (i.e., the collection of daemons, code segments, data stores, and analysis tools that support the generation, storage, and analysis of audit data) and, more specifically, the implementation of the audit record generation process from the moment that an auditable action is initiated to the moment that the audit record is stored in the audit trail.

Access Mediation

If the system contains an access mediation logical subsystem, provide a functional description of the mechanisms that enforce the system security policy. Do not focus on the particular policy enforced by these mechanisms, as policy issues are discussed in the POP. Generally, the access mediation mechanisms base their decision on supplied security attributes (e.g., user identifiers, object names) and stored security information (e.g., attributes contained in databases) associated with the resources protected. Provide pointers to the detailed design documentation that describes the design of the access control mechanism, including how the attributes and security information are defined and managed and how the individual policy enforcement mechanisms are implemented using these attributes.

Privileges and Authorizations

Rights to perform certain system actions may be associated with users or they may be rights exercised by a process or given to a process. These rights are often called privileges or authorizations. Some systems may

have both privileges and authorizations, each of which may be associated with very different types of system actions. If the candidate product implements privileges, authorizations, or a similar concept, discuss the general use of these privilege concepts, noting explicit privilege vectors, and privileged user and user group identifiers. In this section, provide pointers to the detailed design documentation that describes how the privilege concept is implemented. Provide pointers to the detailed design documentation that identifies and describes each privilege and/or authorization, including the circumstances under which it can be granted and to whom (or what) it can be granted, and the implications of its abuse or misuse.

Trusted Processes

In this section of the ASD, summarize the functions provided by each trusted process, noting which software privileges, if any, are necessary to obtain the functionality described. Include a discussion of all security-relevant properties of the trusted process (e.g., it makes security decisions, it is trusted to enforce the system security policy, it is used for administrative purposes) that make it a part of the TCB. Describe the mechanisms used to prevent trusted processes from interfering with each other and the rest of the TCB (e.g., how process isolation is enforced).

Libraries

For each library that is part of the TCB, summarize the functions it provides, noting whether the functionality is influenced by or affected by the use of software privileges. Include a discussion of all security-relevant properties of the library (e.g., it makes security decisions, it is trusted to enforce the system security policy) that make it a part of the TCB. Identify whether TCB software makes use of the library and describe any special aspects of the library that are restricted to TCB use.

This page intentionally left blank

Chapter 3

The TCB Interface Summary Document

The ISD provides summary information on the externally visible interfaces to the TCB. These interfaces are the mechanisms by which users (and subjects acting on behalf of users) obtain services from the TCB. In this section, the required organization of the ISD is defined and relevant concepts and techniques for identifying and classifying TCB interfaces are explained.

The information in the ISD is used in several ways. First, as with all the documents defined in this report, the ISD is crucial to the process that identifies when a product is ready to begin evaluation. Second, the security testing effort, by both the vendor and the evaluation team, uses the ISD as the starting point for a comprehensive security test plan. Finally, the interface summary is used as a general reference by the evaluation team throughout the product evaluation.

The ISD consists of two parts: the TCB Interface Overview and the TCB Interface Summary Tables. The TCB Interface Overview provides a short narrative overview of all TCB interfaces. It describes how TCB interfaces are defined and how security-relevant TCB interfaces are identified. The overview discusses groups of interfaces, possibly with individual examples, as opposed to systematically covering all interfaces in a subsystem. The TCB Interface Summary Tables provide information at the level of individual interfaces. Each individual security-relevant interface is summarized according to a fixed list of required information. The recommended details on these two parts are provided in the next two sections. The only format requirement for the ISD is that the Interface Summary Tables must be included.

3.1 TCB Interface Overview

The TCB Interface Overview begins with a description of how the TCB interfaces are defined and organized. The identification of the TCB (found in the ASD) must be complete before starting the process of identifying the interfaces to the TCB. TCB interfaces are grouped based on logical subsystem. The principle interface types are usually presented in a figure and described in text.

3.1.1 TCB Interface Definition[1]

The TCB interfaces of a product typically consist of (1) user interfaces, (2) program interfaces, and (3) machine/processor interfaces (i.e., processor instructions). These interface categories are likely to have overlap and they ¡include interfaces created by network connections and client/server architectures. The user interfaces include commands that can be input via devices such as keyboards, track balls, and light pens. The program interface consists of all the TCB calls that an application program can make, to include the signal, trap, and fault interfaces that an application program may invoke. The machine/processor interfaces consist of all instructions that refer to TCB internal data structures (e.g., memory registers, segment and

[1] This section is based on material written by the Federal Criteria Working Group.

page tables) and processor registers (e.g., process status registers, segment, page table, capability, cache, and address-translation registers).

Not all commands, system calls, or instructions may, in fact, represent a TCB interface. Some commands and library calls may refer only to programs and data structures that are outside the TCB and hence they are not TCB interfaces. Similarly, some command and application program interfaces may overlap and not represent distinct TCB interfaces. For example, two distinct command interfaces that are implemented by command processors running in user space may invoke the same application program interfaces of the TCB. Consequently, the two distinct commands do not provide distinct TCB interfaces. In some products, the TCB includes the entire application and presents a command interface to its users with no distinct application program interface or processor interface.

3.1.2 Security Relevance

The security-relevant TCB interfaces are the TCB interfaces that:

- Change the security state of the product
- Permit an object access or information flow that is regulated by the security policy
- Are restricted to subjects with privilege or behave differently when executed by subjects with privilege
- Implement or support a security mechanism.

The security state of the product is represented by the objects under the control of the TCB and the attributes of those objects. An interface permits an information flow when it returns information to the calling subject or causes information to flow from one object to another in the product. The key fact about security-relevant interfaces is that one can generate testable assertions about their behavior from the security policy.

3.1.3 Describing Basic TCB Interfaces

Most operating systems products contain basic TCB interfaces that can be presented in a layered description. Figure 3.1 shows the basic TCB interfaces in a UNIX operating system, while Figure 3.2 shows a similar group of interfaces for the IBM VM/CMS environment. In both figures, the hardware instruction set is shown as the most primitive TCB interface. This is appropriate in the case of products in which it is technically possible for users to directly cause the execution of unprivileged hardware instructions. Also included in the hardware instruction group are privileged hardware instructions that may be executed by a privileged user or subject.

After the hardware instructions, the operating system kernel or the Control Program provides the next group of interfaces. In the IBM example, the Conversational Monitor System (CMS) provides a distinct layer of interfaces. In both examples, the uppermost layer of interfaces are provided by trusted programs.

In the UNIX example, system libraries are located in a dashed box near the TCB boundary. This illustrates that libraries may or may not be part of the TCB, depending on a number of factors. These factors include whether the libraries are relied upon by privileged processes and whether they perform security-relevant actions. For any particular UNIX system, a decision is made as to which libraries are in the TCB. This

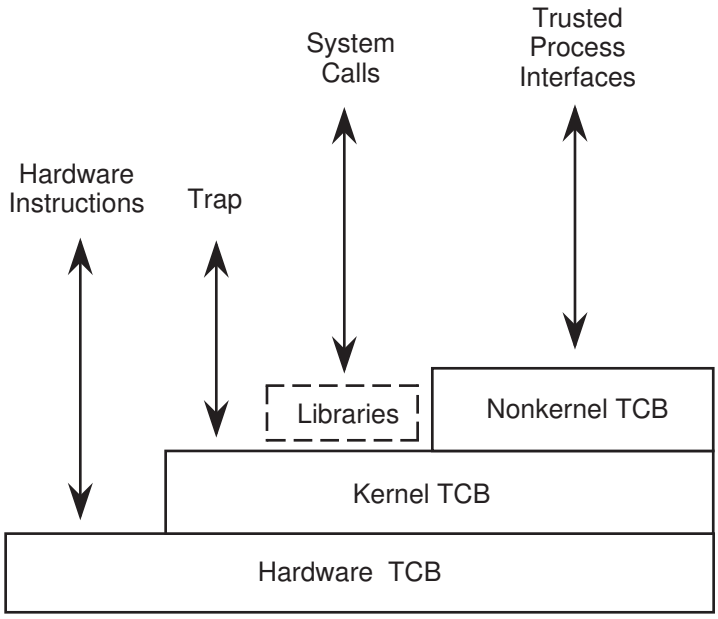

Figure 3.1. UNIX TCB Interface Structure

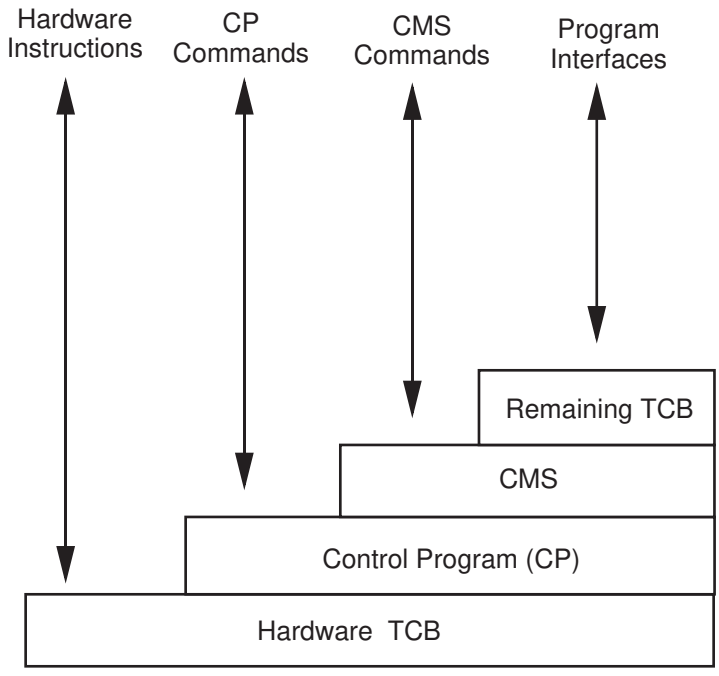

Figure 3.2. IBM VM/CMS TCB Interface Structure

decision should be represented in the diagram whenever possible. The trap instruction is included in the diagram to represent the direct kernel interface.

The examples in this section have been drawn from simple operating systems without the complication of client-server or network interfaces. These more complicated interfaces are discussed next.

3.1.4 Additional TCB Interfaces

A trusted product should be evaluated in a configuration that includes the interfaces required to meet the product mission. This may include network interfaces, client-server interfaces, and external trusted application interfaces. If certain product interfaces are to be excluded from the trusted product, these interfaces should be mentioned in the interface summary along with a brief statement of why they are considered inappropriate for the trusted product as well as how they are excluded from the trusted product.

Trusted Application Interfaces

A trusted application interface (TAI) is a privileged interface that is intended for use with an application that is trusted with the capability to bypass the security mechanisms of the product. A typical example of a trusted application is a database management system. Secure database systems typically enforce MAC and DAC at a level of granularity that is finer than the operating system. To do this efficiently, they need privilege to bypass the OS Mandatory Access Control (MAC) and Discretionary Access Control (DAC) on files.

It is clearly difficult to predetermine the exact range of services that trusted applications may require of an operating system product. Operating system vendors are nonetheless encouraged to identify TAIs in their products whenever possible. This is especially important when the TAIs are internal TCB interfaces of the operating system and would otherwise not be documented and tested in evaluation. The evaluation of trusted application products can be accelerated if the interfaces that they use are documented and tested during the evaluation of the base operating system. Proper evaluation of these interfaces will also make products more attractive to potential customers and system integrators.

Network Interfaces

Network protocol software, such as TCP/IP, is often included in an operating system. The lower-level portions of protocol software often run in a privileged mode. If network protocols are included in the evaluated configuration of a candidate product, then the portion of the network software that is in the TCB is described in the ASD, and TCB interfaces to the networking software must be defined in the ISD.

If network logins are permitted in the evaluated configuration of a candidate product, then the TCB interface to remote users must be described. If network connections are limited to services such as electronic mail, then significantly less interface information is involved.

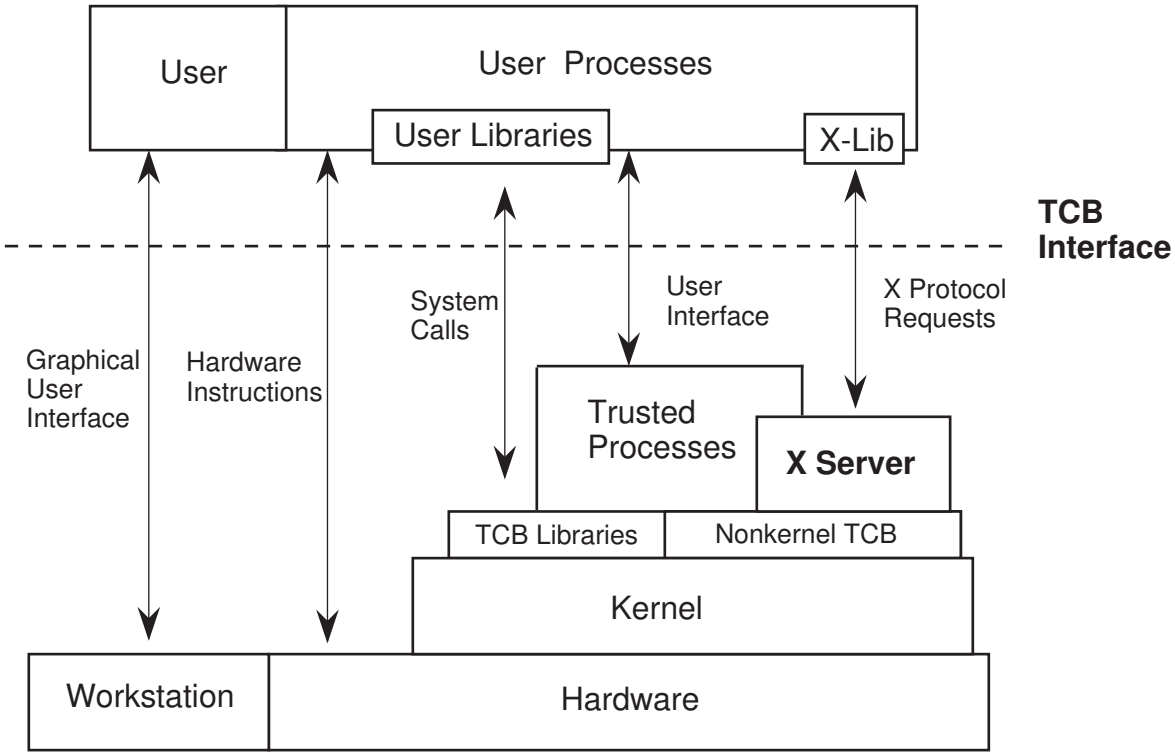

Figure 3.3. Integrating X11 Windows as a TCB Interface

Client-Server Interfaces

An example of client-server interfaces is provided by the X Window System (also called X or X11).[2] X is frequently the foundation of the graphical user interfaces (GUIs) found on UNIX workstations. In a B1 product, a GUI might be employed to display windows of differing sensitivity levels on the same display device. In this situation, the primitive windowing software must be modified to support security enforcement, and it represents a new type of security relevant TCB interface. The situation is pictured in Figure 3.3. The Compartmented Mode Workstations (CMWs) employ the X Window System in this fashion. The detailed design documentation in cases such as these must fully cover the internals of the windowing software.

If a GUI is restricted by a candidate product to operate only in single-level, single-user mode, then the security enforcement of the interface is reduced. In the restricted case, the role of the window system in policy enforcement can likely be documented without full details on the internals of the windowing software.

Processor Interfaces

A trusted product may contain auxiliary processors such as front-end consoles, maintenance or service processors, and I/O processors. These auxiliary processors may be commercial products (workstations or personal computers) or specialized components with their own custom-designed hardware and operating

[2] X Window System is a trademark of the Massachusetts Institute of Technology.

system. All such processors in a product are identified in the ASD. If an auxiliary processor is not part of the TCB, then the interface that the TCB supplies to it must be documented, even if the interfaces in question are restricted to trusted users.

3.2 TCB Interface Summary Tables

The second part of the ISD is the TCB Interface Summary Tables. This section contains one table for each logical subsystem identified in the ASD. Each individual security-relevant interface is listed in the table. Nonsecurity-relevant interfaces may be listed individually or in groups if there is a pointer to a document that describes the group. The format of a table is given in Table 3.1.

Logical Subsystem						
Name	Type	Brief Description	SR	Priv.	Adm.	References

Table 3.1. Format of Interface Tables

The list below describes the meaning of the table entries.

Logical Subsystem The name of the logical subsystem to which the interfaces belong.

Name The name of the interface.

Type The interface type, such as system call, hardware instruction, etc. The interfaces types may vary depending on the candidate product.

Brief Description This entry explains the primary functionality of the interface.

Security Relevant (SR) This column is optional. If nonsecurity-relevant interfaces are included in tables, then this column specifies whether the interface is security-relevant or not.

Privileged (Priv.) This column is used to indicate whether the interface behaves differently when executed with software or hardware privilege. In some cases, it may be useful to identify a privilege in the entry (such as **root**), but this is not required. External trusted application interfaces fall into this category.

Administrative (Adm.) This column specifies whether the interface is intended by the product vendor to be used for administrative purposes.

References The references identify the location of detailed design documentation for the interface.

A very simple example of an interface table is given in Table 3.2. The example addresses process management in a UNIX operating system. The reference column contains template entries of the form **X, p. ?; Y, p. ?**. In an actual vendor table, the letters X and Y would be replaced by document names and the question marks would be replaced by page numbers or page ranges. Two references are shown to illustrate that the detailed information need not reside in one location. The number of references required may vary.

One part of the detailed interface specification for a UNIX command is the user manual documentation or "man page." For security evaluation, an additional reference is usually required that states the subject, objects, and security attributes that are affected by the command and the audit data generated by its use.

Process Management						
Name	Type	Brief Description	SR	Priv.	Adm.	References
_exit()	System Call	Terminate a process.	Yes	No	No	X, p. ?; Y, p. ?
execve()	System Call	Execute a file.	Yes	No	No	X, p. ?; Y, p. ?
fork()	System Call	Create a process.	Yes	No	No	X, p. ?; Y, p. ?
sigvec()	System Call	Signal a process.	Yes	No	No	X, p. ?; Y, p. ?
wait()	System Call	Delay a process.	Yes	No	No	X, p. ?; Y, p. ?

Table 3.2. UNIX Process Management Example

3.3 Example

This section provides an example of interface summary documentation for the VMS[3] operating system. The example covers a portion of I&A at the C2 level. The main area not covered is the I&A interface in the network software.

3.3.1 I&A Interface Summary

The narrative summary of the interface for a logical subsystem includes the types of interfaces and their general purpose. The information needed for our example is described here. The interface summary table, Table 3.3, which is described in the next section, is used to support the narrative summary.

There are three interface types identified for VMS I&A. First, the login command is considered a special type of interface. This is because there is no syntax (other than a carriage return) for the user to initiate login. Second, the Digital Command Language (DCL) provides user interfaces for logout and password change and administrative interfaces for setting parameters related to I&A. The final I&A interface type is the AUTHORIZE Utility, which provides administrative interfaces to manipulate the TCB authorization data.

There may be several ways to organize the interfaces in a logical subsystem into types. In this example, login could have been included with the DCL commands. The vendor is free to choose the approach most suited to the candidate product.

3.3.2 I&A Interface Table

The interface summary table for I&A is given as Table 3.3. Additional information is included here to clarify how the table entries were determined. This further description is not required in the ISD document.

The DCL command SET LOGINS establishes a limit on the number of interactive users on this system. This is a privileged administrative command, but it is not security relevant and could have been omitted. The SET HOST command initiates a request for remote login. The SET UIC command changes the user identification of a process. This is a privileged command and there is no mechanism for an unprivileged user to change the user identification of a process.

[3] VMS is a trademark of Digital Equipment Corporation.

The AUTHORIZE utility is a collection of interfaces that manipulate system authorization files. The utility can be run by an unprivileged user, but system privilege is needed to save results to the system authorization files.

References that are likely to be used in the table are the VMS System Manager's Manual and VMS Authorize Utility Manual. The information in the detailed references must explain how the interface changes the security state of the product.

Identification and Authentication						
Name	Type	Brief Description	SR	Priv.	Adm.	References
login	login	Initiate interactive session.	Yes	No	No	X, p. ?; Y, p. ?
LOGOUT	DCL	Terminate session.	Yes	No	No	X, p. ?; Y, p. ?
SET LOGINS	DCL	Set login limit.	No	Yes	Yes	X, p. ?; Y, p. ?
SET PASSWORD	DCL	Change user password.	Yes	No	No	X, p. ?; Y, p. ?
SET UIC	DCL	Change process user-id.	Yes	Yes	No	X, p. ?; Y, p. ?
ADD	AUTH.	Add a record.	Yes	Yes	Yes	X, p. ?; Y, p. ?
COPY	AUTH.	Duplicate an entry.	Yes	Yes	Yes	X, p. ?; Y, p. ?
CREATE	AUTH.	Create authorization file.	Yes	Yes	Yes	X, p. ?; Y, p. ?
DEFAULT	AUTH.	Set default authorizations.	Yes	Yes	Yes	X, p. ?; Y, p. ?
EXIT	AUTH.	Exit AUTH. subsystem.	Yes	Yes	Yes	X, p. ?; Y, p. ?
GRANT	AUTH.	Grant identifier to user.	Yes	Yes	Yes	X, p. ?; Y, p. ?
HELP	AUTH.	Lists command summaries.	No	No	No	X, p. ?; Y, p. ?
LIST	AUTH.	Write a listing file.	Yes	Yes	Yes	X, p. ?; Y, p. ?
MODIFY	AUTH.	Modifies a record.	Yes	Yes	Yes	X, p. ?; Y, p. ?
REMOVE	AUTH.	Remove a record.	Yes	Yes	Yes	X, p. ?; Y, p. ?
RENAME	AUTH.	Rename a record.	Yes	Yes	Yes	X, p. ?; Y, p. ?
REVOKE	AUTH.	Revoke an identifier.	Yes	Yes	Yes	X, p. ?; Y, p. ?
SHOW	AUTH.	Display authorization data.	Yes	Yes	Yes	X, p. ?; Y, p. ?

Table 3.3. DEC VMS I&A Interfaces

Chapter 4

The Philosophy of Protection Document

The purpose of the POP document is to provide an informal description of the overall design of a system that delineates each of the protection mechanisms employed. The objective is to demonstrate that the mechanisms are adequate to enforce the security policy. This chapter defines the contents of the POP document by presenting a recommended outline. The only format requirement for the document is that it must contain a requirement-by-requirement analysis for compliance to the TCSEC as defined in Section 4.5.

4.1 Subjects/Objects (TCB-Protected Resources)

This section of the document describes the system resources that are protected by the TCB. Generally these resources can be divided into classes or types. The most common classes of resources are users, subjects, objects, and devices. The information to be provided for each of these classes of resources is described in the following sections. Note that if the information outlined below is contained in an informal or formal security model, it does not need to be repeated here; a pointer to the model will suffice.

4.1.1 Users

In most systems, users are external entities that access the evaluated system. This section states in a sentence or two what is meant by a *user* in the context of the evaluated system. It also describes how users' accounts are created and destroyed, for example, through administrator-available commands. The detailed specification for these commands is not provided here, but there is a pointer to the appropriate documentation as described in the ISD. Lastly, this section identifies and provides a brief description (a short paragraph for each attribute) of the security-relevant attributes associated with a user. In general, these attributes are independent of any subject created on behalf of the user and include such things as authentication data, privileges, and clearance. The manner in which these attributes are associated with a user is also described in one or two sentences.

4.1.2 Subjects

A subject is an active entity, generally in the form of process or device, that causes information to flow among objects or changes the system state. This section defines what a subject is in the context of the evaluated system (e.g., a process). This section also describes in a few paragraphs the security-relevant information that is maintained for each subject. Examples of such information include, user identities associated with the process, process privileges, and process sensitivity label. As with users, a very brief description of the manner in which processes are created and destroyed is provided. A pointer to the relevant detailed design discussion is included in the ASD.

4.1.3 Objects

An object is a passive entity that contains or receives information. The TCB controls access to these objects and allows limited information flow between non-TCB subjects. As a general introduction to this section, a definition of an object in the context of the evaluated system is provided. In addition, if there are categories into which objects can be divided, these categories are defined here. In particular, common object distinctions such as named objects and storage objects should be defined if relevant to the candidate product.

Object Determination Approach

This section provides the methodology and rationale to determine the system objects. A suggested approach for performing an object study can be found in [2]. The results of applying the methodology (i.e., the objects identified), as well as a summary of the candidate objects that were eliminated and why, are included here. Examples of commonly identified objects include files, directories, device-related files, interprocess communication-related objects, processes, printer queue entries, and mail messages.

Enumeration of Objects

This section provides a description of the objects identified by the application of the methodology (about a paragraph or two for each object). This includes identification of all security-relevant attributes (e.g., owning user, DAC attributes, and MAC attributes) that are maintained as part of the object and the manner in which these attributes are associated with the object. As part of this discussion of each object, the categories (e.g., named or storage objects defined above) to which the object belongs is noted.

This section does not contain a discussion of the access control policies enforced on these objects. Instead, a pointer to the detailed discussion in the POP (Section 4.2, Statement of Policy) is provided.

4.1.4 Devices

This section details the classes of devices (e.g., disks, tapes, and printers) supported by the evaluated system. This discussion includes a brief description of the device, the security-relevant attributes associated with each device, and the manner in which that association is maintained.

4.2 Statement of Policy

This section describes the system security policies that are enforced by the TCB. The policies considered are the access control policies that serve to restrict access of subjects to objects, integrity policies, and any other system security policies that are enforced by the TCB. The implementation of the mechanisms that enforce these policies is not discussed here, but a pointer to the relevant discussion in the ASD is provided. Note that if the information outlined below is contained in a security model, it does not need to be repeated here; a pointer to the model will suffice.

4.2.1 General Policy Enforced

The section begins with a description of the general security policy that is enforced. As part of the introduction for B1 class products (and C2 class products if an informal model already exists), a pointer to the informal security policy model axioms that dictate control of access to objects by subjects is included. This is followed by a more detailed discussion of the individual security policies.

As an example, a detailed discussion of the MAC policy includes the following:

- A general statement of the policy (including a definition of dominance)
- A description of the types of access that are supported (e.g., read, write, and execute); and, for each object, what actions map into these accesses (e.g., read access to file system objects maps into reading and executing the file and searching a directory); if an object is protected by less than the full MAC policy, an explanation of why is included
- The dominance comparison required (e.g., for a subject to read a file system object, the subject's label must dominate the object's label) and any privileges that allow the bypass of the dominance check
- A description of how the access rules just discussed support the corresponding MAC axioms defined in the informal security policy model
- A description of the algorithm that is used to perform the MAC check when a subject attempts to access an object for a particular mode of access, including any checks for privilege that are performed.

Also, as an example, a sufficiently detailed discussion of the DAC policy includes the following:

- A general statement of the policy
- A description of the types of access that are supported (e.g., read, write, and execute); and, for each object, what actions map into these accesses (e.g., read access to file system objects maps into reading and executing the file and searching a directory); if an object is protected by less than the full DAC policy, an explanation of why is included
- A description of how the access rules just discussed support the corresponding DAC axioms defined in the informal security policy model
- A description of the algorithm that is used to perform the DAC access check when a subject attempts to access an object for a particular mode of access, including any checks for privilege that are performed.

A detailed discussion of the label policy includes the following:

- A description of the labeling controls for generating human-readable output. This discussion would include:
  - A description of the checks that are made with the sensitivity label of print request against the sensitivity label range of the print queue and printer
  - A description of the information included on the banner page of a printout and whether any of the information can be suppressed or altered

  - A brief description of the labeling information included on the internal pages of the printout and whether this information can be suppressed or altered
  - A description of how banner page spoofing is prevented.
- A description of the policies related to labeling disks and tapes used for import and export. A discussion of this would include a description of what label the media itself gets as well as what label the data on the media gets.

A detailed description of the object reuse policy describes how the system prevents scavenging of information from all storage objects. In addition to the storage objects already identified in Section 4.1.3, Enumeration of Objects, TCB structures (e.g., the free memory list, data management buffers, process context, caches used to handle file management information) are subject to object reuse. The object reuse algorithms for these TCB structures are described.

4.3 TCB Protection Mechanisms

The primary function of the TCB is the enforcement of the system's security policy. This section describes this enforcement, focusing on the policy aspects. The implementation of the mechanisms that enforce the policy is discussed in the ASD. In addition, traceability between each mechanism and the relevant TCB interfaces involved is provided. If the information noted below is contained in other documents, a pointer to that discussion suffices.

4.3.1 Hardware Protection Mechanisms

The section states the hardware protection mechanisms utilized by the TCB to implement TCB isolation, noncircumventability, and process address space separation. This includes such things as the mechanisms used to protect virtual and physical memory and protection mechanisms inherent in processors. The detailed discussion of these mechanism is contained in the ASD.

4.3.2 Software Protection Mechanisms

This section describes any software mechanisms utilized by the TCB to implement TCB isolation, noncircumventability, and process address space separation.

This section also describes how the access controls described in Section 4.2, Statement of Policy, are applied to each object protected by the TCB. For each object, information is given on:

- How the object obtains its initial security attributes
- What checks are made when the object is accessed (i.e., MAC, DAC, and privilege checks)
- When the access checks are made (e.g., when the object is initially introduced into the user space (i.e., open()) or every time the object is accessed (i.e., on every read or write))
- What controls are placed on changing object security attributes

- Why the access controls placed on the object are sufficient and consistent with the general security policy.

Additional information includes a discussion of the point in time (e.g., immediately or when there are no references to the object) at which revocation of DAC and MAC access to objects takes place and the point in time at which deletion of objects takes place.

4.3.3 Identification and Authentication Mechanisms

This section provides a description of the various policies that apply to user identification and authentication in interactive and batch processing modes, alteration of session attributes, and creation and use of passwords.

For interactive processing, the following information is included:

- What constitutes a successful logon
- A very brief description of the logon process[1]
- What access control checks are performed after successful authentication (e.g., checks against the user's clearance range and checks against the clearance range of the terminal port device where the user is located)
- What constitutes a logon failure and what events take place on consecutive logon failures.

The ways in which a user can alter the logon session attributes are described. This includes the user's ability to change identity, the user's ability to change group membership, and the user's ability to change the sensitivity level at which the user is operating. Included in the discussion are any access control and identification and authentication policy requirements that must be met to accomplish these changes (e.g., to change membership from one group to another, the user must be a member of the target group; to change user identity, the two user identities involved must have comparable clearances).

This section includes a discussion of the features provided to ensure that authentication data (e.g., passwords) is not compromised. This includes the following:

- Where authentication data is stored and who has access to it
- Time restrictions employed in the management of changing authentication data (e.g., a minimum time period in which users are prevented from changing their authentication data, and a maximum time period by which users must change their authentication data)
- How authentication data can be created and by whom (e.g., passwords can be created by users through a trusted command; passwords are automatically generated for the user)
- Restrictions employed in the creation of authentication data (e.g., minimum and maximum length, checks for palindromes, and checks for user or group names)
- Estimate of the password space size.

[1] The underlying mechanisms and databases used to implement user identification and authentication are described in the ASD.

If there are special (trusted) users or groups of users with specific identifiers (e.g., administrative users) who are allowed to perform special (trusted) functions, this is the place to describe these user roles. The discussion includes how these users are identified and what special capabilities they have.

For the identification and authentication mechanisms associated with batch processing, the following information is provided:

- The manner in which the user's identity and operating sensitivity level are associated with the batch job
- The rules that apply for accessing batch files
- The rules that apply for submitting a batch job.

4.3.4 Audit Mechanisms

This section describes the policy implemented by the auditing subsystem. This includes the following information.

- Audit data storage

  Identifies all forms in which audit data is stored and all places on the system in which audit data is stored.

- Audit data protection

  Describes what access controls are provided that prevent non-TCB processes from accessing audit data (e.g., who owns the audit files and what DAC and MAC attributes are placed on the audit files?)

- Audit coverage

  Briefly mentions the sources from which audit records are generated and the coverage these sources provide. The manner in which audit records are generated is discussed in the ASD.

  The information that is recorded in each audit record is stated. The events that are audited are enumerated. If there are several audit record formats, each one is discussed, including which audit events map to which formats, or a reference is given to this information.

- Audit data analysis

  Describes any commands that are provided to perform pre- or post-audit data reduction for any audit log. If unique audit data from the same period of processing is distributed across more than one audit log, the commands for performing reconciliation are described.

- Audit recovery

  Describes measures incorporated into the audit subsystem to aid in recovering from error conditions and limit the loss of audit data. Include a description of the analysis that occurred to determine the maximum amount of audit data that may be lost. Some error conditions to consider include: audit file exhaustion, system panic conditions, and complete system failure.

4.4 Assurances

One of the key requirements of a trusted system is the assurance that the system properly enforces its policy (and accountability) requirements. This assurance provides the basis for trusting that the protection mechanisms of the system work as advertised. This section describes the assurance features provided by the system. This discussion can be divided into the following areas:

4.4.1 System Architecture

The features of the system architecture that ensure the integrity (i.e., tamperproofness) of the TCB, such as, at a lower level, a two-state architecture and page protection; and, at a higher level, DAC and MAC mechanisms used to protect trusted programs and TCB data files, are described. Additional assurance may be provided by the informal security model and is discussed here. The purpose of the informal security model is to describe the TCB is such a way that one can argue that the security-relevant TCB operations preserve the secure state of the system. In that sense, the informal model serves as a high-level specification that documents the expected constraints and outcomes of the TCB operations.

4.4.2 System Integrity

A description is given of the mechanisms that allow for verification of the correct functioning of the hardware and firmware upon which the TCB relies to enforce the security policy. System diagnostic testing is one such mechanism. The aspects of the hardware and firmware that are tested are delineated. An argument is provided that all security-critical parts of the hardware and firmware are covered by the mechanisms.

4.5 Requirements

This section summarizes the security features of the system presented above, but in a requirement-by-requirement format. For each TCSEC requirement that the evaluated system is intended to meet, a summary is provided of how the system accomplishes its goal. This should be at the level of detail found in the "Evaluation as an X System" section of a TPEP FER, where X is the target level of the candidate product. It is important that each sentence of each requirement be addressed in the summary. Within the summary for each requirement, references for further information are given to the ASD, ISD, or detailed design documentation.

4.5.1 Requirement Section Examples

The examples contained in the following figures are taken from various FERs. The first example in Figure 4.1 is taken from the FER for Data General AOS/VS [5]. The next example in Figure 4.2 is taken from the FER for HP MPE/VE. The last example in Figure 4.3 is a revised section from the FER for AT&T System V/MLS [3].

Each user is identified to AOS/VS through a unique 1 to 15 character username. The user must also enter a password corresponding to the username to authenticate the user identity. Passwords are 6 to 15 characters. Username and password strings are built from the character set "A"-"Z","0"-"9","?","$","_", and ".". Operators are not required to identify and authenticate themselves to enter the system; therefore, the system console must be physically secured.

By default, EXEC allows five logon attempts after which the console will be locked or the connection broken. The system administrator may specify a different number of logon attempts and whether the console should be locked or the line disabled. When a user succeeds in logging on to the system, the user is assigned a PID and all auditable events are associated with that PID and username.

All passwords are stored in the user profile directory. The access control list for that directory is null so that only processes with the superuser privilege may access the user account information. User profiles are created, modified, and deleted via PREDITOR. Passwords may, at the system administrator's discretion, be stored in an encrypted format.

Figure 4.1. C2 I&A Requirement Example

All information in storage objects released back to the system is inaccessible to other subjects. The file system writes an end-of-file marker to newly allocated files and prevents the subject from accessing any data beyond that marker. The physical I/O to the files is done using blocks. Partially filled blocks are padded with fill characters so that a subject may not read beyond what has been written. Data segments are overwritten by zeroes in both main memory and on disk when they are allocated. This includes data segments used by the system as stacks. Tapes and floppy disks must be degaussed before being reused.

Figure 4.2. C2 Object Reuse Requirement Example

System V/MLS can provide human-readable output in three ways: printed on a line printer via the lp command; displayed in a window on the 630 MTG terminal; and displayed on a "dumb" terminal, such as the AT&T Teletype 4425 or 605 terminals. Printable label names associated with sensitivity levels are specified by the system administrator in the files */mls/levels* and */mls/categories*.

In the event that the output is provided by the use of the lp command, the lp subsystem ensures that banner pages are printed for each job that displays the current label of the user. A job may contain several files at different levels from a single user. The label on the banner and trailer pages of the whole print job will be the current label of the user and will therefore always dominate the label of any file contained in the job. Additionally, top and bottom page labels are provided by default; however, these may be overridden by the user. In the event that these are overridden, the act of overriding them is auditable. The top and bottom labels are not printed if the label is longer than 80 characters. In this case, the top and bottom labels are replaced by the character string "** security label for privilege ⟨priv name⟩ is too long to print **" where name is the name of the user's current operating privilege. This label replacement is auditable.

The 630 MTG terminal labels all windows with their sensitivity labels. The label display is dependent upon the size of the window. The hierarchical portion of the label is always displayed, and as much of the nonhierarchical portion of the label as possible. In the event that the nonhierarchical part of the label is too long to be displayed, the 630 MTG flags the condition in the user's display, and provides an option that allows the user to display the entire label.

By default, the user's current level and privilege are defined as each user's prompt in /etc/profile. However, since users may select their own prompts, they may choose not to display their current level and privilege at all times. Also, if the default PS1 prompt, as defined in /etc/profile, is more than 240 characters long, it is replaced by "⟨label too long⟩ privname $" where privname is the name of the user's current operating privilege. For these reasons, System V/MLS provides a mechanism to display the user's current operating privilege at any time.

Figure 4.3. B1 Labeling Human-Readable Output Requirement Example

This page intentionally left blank

Appendix A
Acronyms

ACL	Access Control List
ASD	Architecture Summary Document
CMS	Conversational Monitor System
CMW	Compartmented Mode Workstation
CP	Control Program
DAC	Discretionary Access Control
DBMS	Database Management System
DCL	Digital Command Language
DMA	Direct Memory Access
ECL	Executive Control Language
ER	Executive Request
FER	Final Evaluation Report
FSO	File System Object
GUI	Graphic User Interface
I&A	Identification and Authentication
IPAR	Initial Product Assessment Report
IPC	Interprocess Communication
ISD	Interface Summary Document
MAC	Mandatory Access Control
NCSC	National Computer Security Center
PAT	Process Action Team
PID	Process ID
POP	Philosophy of Protection
RAMP	Rating Maintenance Phase
SR	Security Relevant
TAI	Trusted Application Interface
TCB	Trusted Computing Base
TCSEC	Trusted Computer System Evaluation Criteria
TPEP	Trusted Product Evaluation Program

This page intentionally left blank

Appendix B

Bibliography and References

[1] Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.

[2] Faigin, D. P., Donndelinger, J. J., Jones, J. R., "A Rigorous Approach to Determining Objects," *Proceedings of the Ninth Annual Computer Security Applications Conference*, IEEE Computer Society Press, December 1993, pp. 159–168.

[3] *Final Evaluation Report of American Telephone and Telegraph, System V/MLS Release 1.1.2 Running on UNIX System V Release 3.1.1*, CSC-EPL-89/003, October 18, 1989.

[4] *Final Evaluation Report, CONVEX Computer Corporation, ConvexOS/Secure*, CSC-EPL-92/004, May 1992.

[5] *Final Evaluation Report of Data General Corporation, Advanced Operating System/Virtual Storage (AOS/VS)*, CSC-EPL-89/001, February 22, 1989.

[6] *Final Evaluation Report, HFS Incorporated, XTS-200*, CSC-EPL-92/003, May 1992.

[7] *Final Evaluation Report of Hewlett Packard Computer Systems Divsion, MPE/VE*, CSC-EPL-88/010, October 5, 1988.

[8] *Final Evaluation Report, International Business Machines Corporation, MVS/ESA Operating System*, CSC-EPL-90/002, September 1990.

[9] *Final Evaluation Report, SecureWare, Incorporated, Compartmented Mode Workstation Plus*, CSC-EPL-91/002, January 1991.

[10] *Final Evaluation Report of Unisys Corporation OS 1100*, CSC-EPL-89/004, September 1989.

[11] *Final Evaluation Report, Verdix Corporation, VSLAN 5.0*, CSC-EPL-90/001, July 1990.

[12] *Final Evaluation Report, Wang Laboratories, Incorporated, SVS/OS CAP 1.0*, CSC-EPL-90/004, September 1990.

[13] *A Guide to Understanding Design Documentation in Trusted Systems*, National Computer Security Center, NCSC-TG-007, Version 1, 2 October 1988.

[14] *A Guide to Understanding Security Modeling in Trusted Systems*, National Computer Security Center, NCSC-TG-010, Version 1, October 1992.

[15] Mitchell, H. J., ed., *32-Bit Microprocessors*, McGraw-Hill, 1986.

[16] *Trusted Product Evaluation Questionnaire*, National Computer Security Center, NCSC-TG-019 Version 2, 2 May 1992.

[17] *Process Action Team Guidance Working Group, Form and Content of Vendor Test Documentation*, National Computer Security Center, May 1994.