System Assurance

The aim of system assurance is to verify that a system enforces a desired set of security goals. For example, we would like to know that a new operating system that we are developing can protect the secrecy of one group of users' data from another group. First, we would like to know that the operating system mechanisms (e.g., reference monitor) and policies (e.g., multilevel security) are appropriate to enforce the goal. Second, we would like to know whether the operating system implementation correctly implements intended mechanisms and policies. System assurance describes both what determines reasonable goal and what is a satisficing implementation, and system assurance also describes how a secure operating system should be built and maintained.

The development of Multics, see Chapter 3, and the subsequent security kernel systems, see Chapter 6, demonstrated to the security community that building a secure operating system was a difficult undertaking. The security kernel approach emerged as the appropriate technique for constructing secure system, see Section 6.1. Three specific tasks were identified. First, the system development team must clearly define the security goals for their system. Second, the system development team must construct a system design in such a way that its security properties can be verified, formally if possible. Third, the system development team must implement the kernel in a manner that can be traced to the verified design, automatically if possible. This approach to system development motivation the construction of several design verification tools (e.g., [48, 118, 79, 78, 331]).

Using the security kernel approach as a guide, the US Department of Defense (DoD) developed a set of standards for the security requirements of systems and evaluation procedures for verifying these requirements, called the Rainbow Series [222]. These standards cover security function ranging from passwords and authentication to recovery and audit to access control and system configuration. Further, there are standards for procedures including documentation, procurement, facilities management, etc. The standards covering operating systems, called the Trusted Computer System Evaluation Criteria (TCSEC) or the “Orange Book” [304], have generated the most discussion and probably the most use. The Orange Book defines a progressively-more-secure sequence of requirements for operating systems. The Orange Book combines desired security features with the tasks to verify correct implementation of those features into a set of assurance classes. Ultimately, this approach was found to be unnecessarily constraining, as the levels of security function and assurance may not coincide. In the succeeding assurance approach, called the Common Criteria [60], evaluation criteria are organized into distinct components: the security function into the target of evaluation (TOE) and evaluation effort into the evaluation assurance levels (EALs). Thus, a system's particular security features can be assured to different levels depending on its design, development, documentation, configuration, testing, etc.
In this chapter, we describe the Orange Book and Common Criteria assurance methodologies and their impact on operating system design and implementation.

12.1 ORANGE BOOK

The TCSEC or Orange Book was developed by the US DoD’s Computer Security Center which was formed in 1981 [304]. The main focus of the center was to encourage the development of secure operating systems by vendors. However, as discussed throughout the book, building a secure operating system is a challenging task, and by this time the commercial market had largely moved away from the features of secure operating systems (e.g., Multics [237] and security kernels, such as Scomp [99] and GEMSOS [290]) to the function and flexibility of UNIX systems which did not focus on security (see Chapter 4). Thus, the center needed to not only define how to build such systems, but also encourage the development of secure operating systems.

The Orange Book defines two sets of requirements: (1) specific security feature requirements and (2) assurance requirements. The specific security feature requirements define the operating system features that are necessary to enforce the security requirements. The assurance requirements specify the effort necessary to verify the correct implementation of the specific security features.

The Orange Book is primarily known for its definition of assurance classes that dictate specific security feature and assurance requirement combinations. Ultimately, these classes were organized into divisions with similar assurance requirements (A through D). Each class defines four categories of requirements for their systems: (1) the security policy model, including the administration of policies described in the model and the labeling of system resources; (2) the level of accountability for system actions, including authentication of individual subjects and audit of system actions; (3) the degree of operational assurance that the system behaves as expected, including the implementation and maintenance of the system; and (4) the documentation provided to support the design, implementation, assurance, and maintenance of the system. The first two categories describe the specific security feature requirements that dictate the security function intended by the system design. The second two categories specify assurance requirements that determine whether the implementation satisfies the intended design. Each class specifies its minimal requirements for these categories.

We describe Orange Book classes below, starting with the lowest security classes (i.e., least assurance of security enforcement).

D – Minimal Protection  Class D is reserved for systems that have been evaluated, but fail to meet the requirements of any higher security class.

The next set of classes support discretionary access control security policies. These classes are grouped into a division called Discretionary Protection (division C) where only discretionary mechanisms are necessary to evaluated be at one of these classes. Other features and assurance requirements distinguish the classes.
C1 – Discretionary Security Protection  A Class C1 system requires a discretionary access control (DAC) model specifying the permissions of named users to named objects. Users must authenticate themselves prior to performing any system actions. System actions are performed by hardware-protected domains whose rights are associated with the authenticated users. Assurance requires testing that there are no obvious ways to bypass these controls. Basic documentation is required.

This class defines a basic DAC system with hardware protection of processes and user authentication.

C2 – Controlled Access Protection  A Class C2 system provides the DAC model of Class C1 where the rights may be specified at the granularity of a single user, and where administration is authorized. Authentication shall be based on a secret (e.g., a password) that is protected from access by other users. Audit and object reuse are introduced by this class. Audit of a specific set of events is defined, and such auditing requires a protected log. Object reuse means that when an object is provided, such as a file, no data from its previous use is accessible. Assurance requires testing for obvious flaws and obvious bypass. Documentation for users, facilities, design, and testing is required.

C2 is the evaluation level for most discretionary systems, such as Windows and UNIX systems, historically. Access control is based on a user identity, and passwords are protected from access. As of 1996, five operating systems were assured at this class, including Windows NT 3.5 Service Pack 3, albeit without networking.

The next division, called Mandatory Protection, defines classes of systems that provide mandatory access control (MAC).

B1 – Labeled Security Protection  A Class B1 system provides the DAC, audit, and object reuse features of Class C2 plus a MAC model where each named subject and object is associated with a sensitivity label, corresponding to a multilevel security (MLS) policy (see Section 5.2). These labels must be integrity-protected and exported with the data when it leaves a system (e.g., by device or via a printed page). Authentication must identify a user by sensitivity level for authorization. Assurance now requires that the security mechanisms of the system must work as claimed in the system documentation. An examination of the design documentation, source code, and object code are necessary to prove this. The documentation must support such testing through detailed descriptions of the security policy model and protection mechanisms, including a justification how they satisfy the model.

B1 is the class in which mandatory access control is introduced. Also, at this stage detailed testing that the documentation and source code implement the intended security features is required. Seven operating systems were evaluated at this class, including Trusted Solaris V1.1 Compartmented Mode Workstation (see Chapter 8).
B2 – Structured Protection  A Class B2 system extends the B1 class by requiring enforcement on access to all subjects and objects (i.e., not just named ones) and covert channel protections (see Section 5.4). At this stage, authentication requires a trusted path (see Section 7.5). Also, requirements on the trusted computing base (TCB) design are added. The protection-critical part of the TCB must be identified, and its interface must be well-defined. At this point, the TCB must be shown to be “relatively resistant to penetration.”

B2 is the class at which covert channels are first mentioned. This introduces a new significant and complex design and evaluation task to the assurance process. The evaluation of the TCB is also much more detailed. The design specification includes a “descriptive top-level specification” that includes an accurate description of the TCB interface and its exceptions, error messages, and effects. Only two systems were evaluated at this class, Trusted Xenix 3.0 and 4.0 [111].

B3 – Security Domains  A Class B3 system extends the B2 class by requiring that the TCB satisfy the reference monitor concept in Definition 2.6. The TCB design and implementation are directed toward minimal size and minimal complexity. The system is expected to be “highly resistant to penetration.” At this point, the audit subsystem must be able to record all security-sensitive events.

At this point, we see the requirements for a secure operating system, but without the formal verification necessary for Division A assurance below. As of 1996, only one system was assured at B3, the XTS-300 system [21] that was a successor to the Scomp system discussed in Section 6.2.

A1 – Verified Design  A Class A1 system is functionally equivalent to a B3 system, but the evaluation of this system must be derived from a formal design specification. Unlike the other classes, the assurance of a Class A1 system is developmental in that the design and implementation of the system follow from a formal top-level specification (FTLS).

A Class A1 system must meet the following five requirements:

1. A formal model of the security policy must be documented and include a mathematical proof that the model is consistent with the policy.
2. An FTLS must specify the functions that the TCB performs and the hardware/firmware support for separate execution domains.
3. The FTLS of the TCB must be shown to be consistent with the formal model of the security policy.
4. The TCB implementation must be consistent with the FTLS.
5. Formal analysis techniques must be used to identify and analyze covert channels. The continued existence of covert channels in the system must be justified.
A Class A1 system is a secure operating system that has been semi-formally verified to satisfy the reference monitor guarantees. Building a Class A1 system requires diligence from the outset of the development process to build and verify the formal specifications of the system design (FTLS), the security policy, and the TCB of the system. No Class A1 operating systems are commercially available, although custom systems, such as BLACKER [330] based on GEMSOS [290], have been evaluated at A1, and the VAX/VMM system (see Section 11.2) was designed and implemented to be evaluated at Class A1.

Beyond Class A1 The Class A1 requirements include exceptions that permit informal verification where no formal analysis tool exist. In particular, research in the 1980s indicated that formal verification of security properties down to the level of the source code may be possible, so the Beyond Class A1 left open the possibility that security requirements could be verified more precisely. Formal verification tools that can prove the satisfaction of the variety of security properties required for large source code bases, such as operating systems, have not become practical. As a result, no further classes were defined.

12.2 COMMON CRITERIA

While the Orange Book defined targets for security features and assurance, the linking of features and assurance requirements became cumbersome. A detailed formal assurance may be performed for systems that do not provide all of the features in B3 (e.g., covert channel protection). Also, the six classes of security features themselves were constraining. The market may determine combinations of security features that do not neatly fit into one of these classes. Out of this desire for a more flexible assurance approach, the Common Criteria emerged [60, 61].

In the early 1990s, independent approaches for system assurance were be developed in Europe and Canada. The Information Technology Security Evaluation Criteria (ITSEC) version 1.2 was released in 1991 as a joint standard used by France, Germany, the Netherlands, and the United Kingdom [147]. ITSEC defines a set of criteria for evaluating a system, called the target of evaluation, to verify the presence of a set of security features and to verify its defenses against a comprehensive set of penetration tests. The set of security features implemented by a system need not conform to a specific TCSEC class, but rather can be defined as part of the evaluation, called the security target. The amount of evaluation effort determines a confidence level in the target of evaluation, called evaluation levels that range from E0 (lowest) to E6 (highest).

In Canada, the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) defined its evaluation approach in 1993 [41, 194]. CTCPEC leverages the security feature requirements of the Orange Book while incorporating the notion of distinct assurance levels of the ITSEC approach.

Inspired by facets of each of these evaluation approaches, the Common Criteria approach was developed [60]. Like the ITSEC and CTCPEC approaches, the Common Criteria separates the assurance effort from the security features being assured. Like the Orange Book, the Common Cri-
teria defines security feature configurations that would lead to a meaningful assurance for developers and consumers.

### 12.2.1 COMMON CRITERIA CONCEPTS

An overview of the Common Criteria approach is shown in Figure 12.1. The Common Criteria consists of four major concepts: (1) the *target of evaluation* (TOE) which is the system that is the subject of evaluation; (2) the *protection profile* (PP) which specifies a required set of security features and assurance requirements for systems to satisfy that profile; (3) the *security target* (ST) which defines the functional and assurance measures of the TOE security that are to be evaluated, perhaps satisfying one or more PPs; and (4) the *evaluation assurance level* (EAL) which identifies the level of assurance that the TOE satisfies the ST. In a Common Criteria evaluation, the TOE’s security threats, security objectives, features, and assurance measures are collected in an ST for that system. The ST may claim to satisfy the requirements of one or more PPs which defines a set of requirements on the ST. The PPs typically indicate a minimum EAL that can be assured using its assurance requirements, but the ST may exceed these assurance requirements to satisfy the requirements of a higher EAL. We examine these concepts in more detail below.

**Figure 12.1:** Common Criteria Overview: A *target of evaluation* is converted to a *security target* that defines the features and assurance being evaluated. The *protection profile* specifies the security features required and the *assurance levels* define a set of possible assurance requirements for those features.
The protection profiles (PPs) are generally derived from the popular TCSEC classes. For discretionary access control (DAC) systems, the Controlled Access Protection Profile (CAPP) was developed that corresponds to TCSEC Class C2 [230]. The functional requirements of a CAPP system correspond to a Class C2 system, and CAPP also defines assurance requirements for a typical system with a minimum EAL 3 assurance required (see below for EAL definitions). For mandatory access control (MAC) systems, the Labeled Security Protection Profile (LSPP) was developed [231]. It corresponds to TCSEC Class B1. LSPP also defines assurance requirements that require a minimum of EAL3 assurance, although a higher level of assurance is expected.

<table>
<thead>
<tr>
<th>Level and TCSEC Map</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAL1</td>
<td>functionally tested</td>
</tr>
<tr>
<td>EAL2 (C1: low)</td>
<td>structurally tested</td>
</tr>
<tr>
<td>EAL3 (C2/B1: moderate)</td>
<td>methodically tested and checked</td>
</tr>
<tr>
<td>EAL4 (C2/B1: medium)</td>
<td>methodically designed, tested and reviewed</td>
</tr>
<tr>
<td>EAL5 (B2: high)</td>
<td>semiformaly designed and tested</td>
</tr>
<tr>
<td>EAL6 (B3: high)</td>
<td>semiformaly verified design and tested</td>
</tr>
<tr>
<td>EAL7 (A1: high)</td>
<td>formally verified design and tested</td>
</tr>
</tbody>
</table>

**Figure 12.2: Common Criteria Evaluation Assurance Levels**

The evaluation assurance levels (EALs) and their abstract test and verification requirements are listed in Table 12.2. The EAL levels indicate identifiable assurance that the ST is implemented correctly by the TOE system. It is important to note that the assurance level does not indicate that a system has more security features, but rather whether the system’s design and implementation assures that the support security features are correct. Both the security features and assurance must be assessed to determine whether the system satisfies the requirements of a secure operating system (Definition 2.5 in Chapter 2).

Table 12.2 also includes an indication of high, medium, moderate, and low assurance for the levels. This categorization is not universally agreed upon, but the important thing to take away is that there are significant assurance differences between the aggregations. For example, EAL3 is the first level to require a methodical testing, including grey box testing and a search for vulnerabilities. Also, a high-level design description is required. However, EAL4 assurance requires a much higher effort on the part of the developer for a low-level design and a detailed vulnerability test. EAL5-7 are all considered as high assurance as the levels all require some application of formal methods to the assurance. Only 2 products on the Common Criteria’s List of Evaluated Products have been evaluated at EAL7 (two data diode devices for restricting the flow of secret data from Tenix) and none have been evaluated at EAL6. For operating systems, only the Processor Resource/System Manager (PR/SM) system from IBM and the XTS-400 system from BAE systems are evaluated at EAL5.
Security Targets (STs) are constructed from the TOE system using a set of predefined components. There are functional components that cover security features, such as for audit, cryptographic support, user data protection (access control), and authentication. There are also assurance components that cover the extent to which the ST features can be evaluated, including configuration management, system delivery and operation (deployment), documentation, and testing. The assurance components also include ones for assuring that PP are satisfied and that the ST itself is complete, consistent, and technically sound.

The components form the foundation of STs and PPs, and as such they are low-level. For example, CAPP uses functional components \([58]\) for User Data Protection for discretionary access control policy (indicated by a component identifier, FDP_ACC.1), discretionary access control function (FDP_ACF.1), and object reuse (FDP_RIP.2). User Data Protection forms a class of requirements, identified by the name FDP. Each of these components that were chosen belong to different families (FDP_ACC, FDP_ACF, and FDP_RIP, respectively). LSPP builds on the CAPP requirements for User Data Protection by adding functional components for mandatory access control (FDP_IFC.1) and label import and export (FDP_ITC.2 and FDP_ETC.2), among others. Thus, the PP definitions consist of a set of functional components organized by their classes.

Similarly, STs would be defined from the same pool of functional components, but they must also be defined in terms of assurance components in order to be evaluated against the specified EAL. Assurance components refer to classes of requirements in the areas of configuration management (ACM), delivery and operation (ADO), life cycle support (ALC), vulnerability assessment (AVA), etc. The EAL definition includes a set of components from these classes.

### 12.2.2 COMMON CRITERIA IN ACTION

Recently, RedHat Linux Enterprise Linux Version 5 was assured at EAL4 for LSPP. What did it take to perform this evaluation? First, the Linux system formed the TOE of evaluation, so an ST for this system was constructed. Building an ST for achieving LSPP at EAL4 requires by collecting all the components required by LSPP \([231]\) and all the components required by EAL4 \([59]\) into a single ST specification. LSPP requires nearly 40 functional requirement components and nearly 20 assurance requirement components. The functional requirement components include the obvious ones for multilevel security (MLS), which is at the heart of the profile, but also components for export/import of labeled data, among others. The assurance requirement components for LSPP include those necessary to describe the system design (a high-level design for LSPP) and manage the deployment and maintenance of the system. As the assurance level does not impact function, EAL4 adds only assurance requirements to those required for LSPP. LSPP is designed to support EAL3 evaluation, so the additional requirements of EAL4 evaluation above EAL3 must be added to the ST. There are 7 additional assurance requirements for EAL4 over EAL3. While this is a small number of components, these require significant additional evaluation effort. These additional components include a complete functional specification, an implementation representation of the system, module-level design, and a focused vulnerability analysis.
Once the Linux ST is defined, the TOE must be evaluated against these requirements. For each of the nearly 70 requirements, a set of elements must be fulfilled. For example, a complete functional specification (ADV_FSP.4) has 10 elements. There are elements related to the development of the system, verification of the system’s content and presentation, and evaluation of the quality of the other elements. First, two elements must be provided by the developer: (1) a functional specification and (2) a description linking the functional specification to all the security functional requirements. A case must be made as to how the Linux TOE satisfies those requirements. These elements define the system structure and how it implements the required security functions.

Next, there are six content and presentation elements that specify how the system satisfies its security requirements. These include: (1) demonstrating that the functional specification completely represents the trusted computing base of the system, called the target of security function (TSF); (2) the purpose and method of each interface in the TSF; (3) a description of all the parameters for each interface in the TSF; (4) all the actions performed based on interface invocation of the TSF; (5) all error messages that may result due to an interface invocation of the TSF; and (6) a description linking the security functional requirements to the interfaces of the TSF. Cumulatively, these describe how the system’s trusted computing base is invoked and how it implements the required security function when invoked.

Lastly, the evaluator must confirm that the specifications above are complete and correct. There are two elements required: (1) the evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence and (2) the evaluator shall determine that the functional specification is an accurate and complete instantiation of the security function requirements.

Note that this is just one of the nearly 70 requirements that needs to be fulfilled as part of the evaluation. While this is one of the most demanding elements, the level of effort necessary to fulfill each of the elements is significant. Many pages of documentation are generated in the assurance process, and resulting costs are impressive.

In the end, the degree of effort taken in an evaluation process should have some positive effect on the quality of the target system. However, even the effort put into verifying the functional specification does not prove correctness in any concrete sense. Notice that the requirements above only require a manual examination of code and an informal description of how the security requirements are implemented.

As EAL4 aims for medium assurance, this level of analysis is appropriate, but it is generally accepted that general-purpose, commercial operating systems, such as Linux, BSD, and Windows, will never be evaluated at a higher assurance level due to their complexity and the limits of their development processes. For higher assurance, the development process must be tightly controlled, such that the formal specifications can be generated and tested against the code. In the future, we need assurance techniques and tools that enable effective development while constructing the necessary case for source code level assurance. These challenges have been with the security community since the evaluation process was proposed, but we still have a ways to go.
12.3 SUMMARY

System assurance is the process of verifying that a system enforces a desired set of security goals. The TCSEC or Orange Book was developed in the early 1980s to define how to verify security in operating systems. The Orange Book defined a set of classes that specified distinct security and assurance requirements for operating systems. The Orange Book notion of system assurance gained acceptance, if only because the US DoD mandated the use of assured systems.

However, the European and Canadian and eventually American security communities saw the need to separate the security function of a system from the assurance that it truly implements that function. The Orange Book notions of security function were extended and combined with the assurance approaches developed in Europe (ITSEC) and Canada (CTCPEC) to create the Common Criteria approach.

Many more evaluations have now been performed using the Common Criteria than were ever performed under the Orange Book, resulting a worldwide approach to assuring systems. Performing a system evaluation is still strongly motivated by government markets, and the cost of performing even medium-level assurance for most systems is prohibitive. Further, the current, commercial development process precludes high assurance of systems. The weak link in assurance has been and continues to be techniques and tools to build systems whose security can be verified in a mostly, automated fashion. While this problem is intractable in general, work continues on developing usable, formal verification techniques (e.g., NICTA [81]).