
I-0453: TSF And Support Code

TYPE: Request for Interpretation
NUMBER: I-0453
STATUS: Reposted for External Review

TITLE: TSF And Support Code

FIRST POST: [\[cc-cmt 00364\]](#)
MOST RECENT REPOST: [cc-cmt TBD]
COMMENTS DUE BY: Friday, February 13, 2004 to cc-cmt@nist.gov

RELATED TO:
[I-0054](#) Support Code Need Not Meet All Architecture Requirements

ISSUE:

Given the current wording of the definition of TSF, all support code is part of the TSF because it is necessary to meet AMT, TST, etc. This, however, implies it is subject to INT, ALC_*, and so on. Is this appropriate for integrity tests, self-tests, etc.?

PROPOSED RESOLUTION

For less critical portions of the TSF, it should be acceptable to use a balanced assurance argument, accepting a lower level of assurance for some identified subset of the TSF. Such an argument must be accompanied by risk assessment that shows the risk due to malfunction of that subset to be lower, thus justifying less assurance work.

SPECIFIC RESOLUTION

To add support for Balanced Assurance, the CC must be modified to incorporate the following notions:

- Threats must be made specific as to the time of attack, e.g., the attacks from external users would occur after the system is operational.
- Objectives must be made specific as to the portion of the code to which they apply (if applicable). For example, an objective regarding protection of user data must indicate that it is effective once the system has reached a secure state.
- The assurance requirements must be iterated to apply to different subsets of the TOE, where applicable. A balance assurance argument must be provided to justify that each subset is given an appropriate level of assurance based on the threats and objectives.

SUPPORT:

In Part 1, Clause 2, the CC v2.1 defines the TSF as the set of all hardware, firmware, and software that must be relied upon for the correct enforcement of the TSP. It defines the TSP as the set of rules that regulate how assets are managed, protected, and distributed within a TOE. From this, one can deduce that the TSF is related to those components critical to the enforcement of protection and management policies. This clearly would apply to SFR from the FIA, FAU, FMT, and FDP classes, and likely also from FCS, FPR, and FTA classes, as these deal with aspects of protection as well.

FPT and the Part 3 assurance requirements are a different beast. For some, it is clear that they must work correctly for protection to work correctly. The best examples of this are FPT_SEP and FPT_RVM.

But what about FPT_AMT? Must the same level of assurance be provided if these tests run only in a maintenance mode, where the general user threats may not be present? Similar arguments would apply for FPT_TST? What about test

scaffolding developed to meet the ATE requirements? Must that meet the same level of assurance? What about initialization code, that runs only when the system is booting and before it is available for normal user operation? Is that code subject to the same risks?

It seems clear that different code faces different risk, depending on the time and environment in which it is executed. This argues for the acceptability of a Balanced Assurance approach.

Balanced Assurance permits lesser assurance for those portions that are less critical. For example, consider code is not executed when there is a concurrent ability for general user access. Although the code still needs to work correctly, there may not be the need for as extensive vulnerability analysis or design analysis, for the risk of attack from the general user population is lower.

In the days of the TCSEC, there was a category of code that was called "Maintenance" or "Support" code. It was explicitly limited to code that could not be invoked as part of the operational TCB; for example:

- *Initialization code*, whose task is to bring a system to a secure initial state, configures available peripherals and initializes TCB data structures.
- *System integrity test code*, which is unavailable once the TCB is operational. Such code might be available only in a restricted maintenance mode.
- *Trusted Recovery support code*, if it is available only in maintenance mode.
- *Installation and TCB generation support*, if it is unavailable once the system is operational.

Other examples, not included in the original TCSEC interpretation, are as follows:

- *Diagnostic Programs*. This is code used to determine what is wrong with the hardware (or software) but which executes in a standalone "maintenance mode", or in some other restricted environment.
- *Backup Utilities* This code is used to create system backups without any users logged-in. On-line backup utilities may not be able to guarantee complete consistency among data files.
- *Recovery Utilities*. After system errors it may not be possible to keep the system running and preserve security. Users may have to be logged off and integrity checks run with the security officer re-labeling objects.
- *Media Utilities*. This code is used to format disks, tapes, and other media. It may need to be performed in a standalone mode due to hardware (or software) limitations.
- *System Generation*. This is code used to generate a new or updated operating system image. It can be done online but is just as likely to require the system standalone.
- *System Load*. This is code used to select a particular boot image.

TCSEC Interpretation I-0054 stated that "support code" had to meet the following requirements:

1. It must be non-invokable through TCB interfaces, and must not be invoked by the TCB, once the TCB has reached secure state.
2. Such code must be protected from external interference or tampering (e.g., by modification of code or data structures).
3. The interfaces and elements categorized as support code needed to be identified at a level appropriate for the digraph.

This is a Balanced Assurance argument. It appears that the permitting of such an argument should be acceptable under the CC.