General-Purpose Operating System Protection Profile

A joint effort by NIAP and BSI for the development of a Common Criteria Protection Profile

Version 3.9

Part2: General Approach and Assurance Activities for OSPP Evaluations

Table of Contents

Introduction

This document presents the ideas for assurance activities tailored to general-purpose operating systems and the basic security functional requirements presented in OSPP part 1. While part 1 is assumed to be stable, this document is still a working draft and produced by collecting a number of white papers and by defining SFR-related assurance activities.

The three white papers included here address the following topics:

- Approach to developer documentation required for the evaluation
- General approach for testing (individual SFR-specific test requirements can be found in the description of the SFR-related assurance activities)
- General approach to vulnerability analysis

They present the overall framework and are followed by assurance activities for the different SFRs.

In order to keep specific topics together, assurance activities for management SFRs have been kept together with the functionality they manage. This allows the analysis and testing of specific functionality to be performed in conjunction with the management functions that define and modify the allowed configuration of the functionality.

The following table shows where to find the specification of the assurance activities for the individual SFRs in the OSPP.

SFR	**Assurance Activities**
FAU_GEN.1	Evaluation Activities for Security Audit
FAU_GEN.2	Evaluation Activities for Security Audit
FAU_SAR.1	Evaluation Activities for Security Audit
FAU_SAR.2	Evaluation Activities for Security Audit
FAU_SEL.1	Evaluation Activities for Security Audit
FAU_STG.1	Evaluation Activities for Security Audit
FAU_STG.3	Evaluation Activities for Security Audit
FAU_STG.4	Evaluation Activities for Security Audit
FDP_ACC.1	Evaluation Activities for User Data Protection
FDP_ACF.1	Evaluation Activities for User Data Protection
FDP_IFC.1	Evaluation Activities for User Data Protection
FDP_IFF.1	Evaluation Activities for User Data Protection

SFR	Assurance Activities
FDP_RIP.2	Evaluation Activities for User Data Protection
FIA_AFL.1	Evaluation Activities for Identification and Authentication
FIA_ATD.1	Evaluation Activities for Identification and Authentication
FIA_UAU.1(RITE)	Evaluation Activities for Identification and Authentication
FIA_UAU.1(HU)	Evaluation Activities for Identification and Authentication
FIA_UAU.5	Evaluation Activities for Identification and Authentication
FIA_UAU.7	Evaluation Activities for Identification and Authentication
FIA_UID.1	Evaluation Activities for Identification and Authentication
FIA_USB.1	Evaluation Activities for Identification and Authentication
FIA_PK_EXT.1	Evaluation Activities for Identification and Authentication
FMT_MOF.1	Evaluation Activities for Identification and Authentication
FMT_MSA.1	Evaluation Activities for User Data Protection
FMT_MSA.3(DAC)	Evaluation Activities for User Data Protection
FMT_MSA.3(NI)	Evaluation Activities for User Data Protection
FMT_MSA.4	Evaluation Activities for User Data Protection
FMT_MTD.1(AE)	Evaluation Activities for Security Audit
FMT_MTD.1(AS)	Evaluation Activities for Security Audit
FMT_MTD.1(AT)	Evaluation Activities for Security Audit
FMT_MTD.1(AF)	Evaluation Activities for Security Audit
FMT_MTD.1(CM)	Evaluation Activities for Identification and Authentication
FMT_MTD.1(NI)	Evaluation Activities for User Data Protection
FMT_MTD.1(IAT)	Evaluation Activities for Identification and Authentication
FMT_MTD.1(IAF)	Evaluation Activities for Identification and Authentication
FMT_MTD1(IAU)	Evaluation Activities for Identification and Authentication
FMT_REV.1(OBJ)	Evaluation Activities for User Data Protection

SFR	Assurance Activities
FMT_REV.1(USR)	*work in progress*
FMT_SMF_RMT.1	*work in progress*
FMT_SMR.1	*work in progress*
FPT_STM.1	Assurance Activity for FPT_STM.1 and FTA_SSL
FTA_SSL.1	Assurance Activity for FPT_STM.1 and FTA_SSL
FTA_SSL.2	Assurance Activity for FPT_STM.1 and FTA_SSL
FTP_ITC.1	Assurance Activities for Trusted Path/Channels

This document is a first draft for those assurance activities and is expected to be completed and refined also with the first trial evaluations that are performed using the OSPP. Especially the list of generic flaw hypotheses is still incomplete and will be refined in the next versions of this document. Also additional details of assurance activities for the SFRs will be added from the experience gathered in the trial evaluations.

The assurance activities described below are intended to be refinements of security assurance requirements as defined in part 2 of the Common Criteria and their related work units in the CEM. A set of assurance components from part 2 of the Common Criteria has been selected that reflects the evaluation assurance activities the authors of this document view as necessary for the evaluation of a general-purpose operating system product. This set of assurance components is not one of the evaluation assurance levels defined in part 2 of the Common Criteria, but consists of assurance components that are covered by the CCRA.

The final goal is to make the evaluations of such complex IT products like general-purpose operating systems as objective and repeatable as possible.

White Paper on Documentation

In order to come to agreement on the assurance activities to be performed by evaluators, we need to agree on the level of information we expect to be provided and what the expectation is on the evaluator on how they would use that information. Our proposed approach is documented in this whitepaper.

We believe there is agreement on what one would expect for Administrative Guidance and User Guidance, so that documentation is not discussed in a comprehensive manner. However, some system elements that are discussed in the documentation described by this paper will have an impact on the contents of the Administrative Guidance; these aspects are discussed below. The documentation of interest has been categorized as interface documentation and architectural design documentation.

It is important to remember that one of the goals of the activities that will be performed using this information is to provide end-users and application writers an unambiguous list of interfaces that have been analyzed and tested by the evaluation team.

The activities that comprise the vulnerability analysis and testing are left to be described in another white paper.

Principles for Requested Documentation

In formulating the requirements for the documentation, we followed several principles that we feel are consistent with the direction that the community has been taking with respect to Common Criteria evaluations. These principles are as follows:

- Large-scale production of CC-specific documentation is discouraged. The evaluation should be founded on existing developer documentation. While it is inevitable that the vendor will have to develop some documentation (Security Target), the intent of the documentation requirements is to minimize this documentation. The developer is encouraged to refer to existing documentation rather than create new documents. Information regarding interfaces directly interacting with evaluated security functionality should be publicly documented. There may be exceptions (e.g. when wrappers form the interface documented but technically are not the interface, the “technical” interface can be unadvertised or its specification considered proprietary), but these should be rare.
- The focus of the interface description to be provided is to allow the specification of security functionality that implements the security requirements in the PP/ST, and to demonstrate to the end users that this functionality has been tested through the advertised interfaces. There may be cases where interfaces indirectly interact with a security function (e.g., providing an interface to an external, supporting entity), those interfaces also need to be sufficiently documented.
- As the use and abuse of both hardware and software privilege mechanisms, and system initialization code, has a significant impact on the overall security of the system, documentation related to the structure and interaction between

the implemented privilege mechanisms and system startup/initialization must be provided as architectural design information.

- As with the concerns regarding privilege, the soundness of the protection mechanisms the product employs plays a critical role in the security posture of the product. The identification of these mechanisms and how they are used must de described in the architectural design.
- An objective of the evaluation is to ensure that the identified interfaces and the described security functionality satisfies the SFRs and behaves as expected – done through analysis and black-box testing. Another objective is to ensure the product's initialization process results in a secure state. While yet another objective is to determine if the protection mechanisms and use of privilege are sufficiently described and verified. During the course of performing their activities relating to these objectives, the evaluator may uncover security relevant interfaces or design aspects left undocumented, but that is not the focus of these activities. The final objective of determining the products resistance to attack by performing a vulnerability analysis is where the evaluator may consider interfaces that were excluded from the set provided by the developer.
- Various approaches can be taken in providing the architectural design. While it is possible for the vendor to provide a trivial decomposition and still conform to the requirements, our expectation is that a more robust decomposition that accurately reflects the system's privilege architecture – and the fact that the evaluators and Schemes will provide public comments on this architecture – will be seen as providing a marketing advantage over the trivial decomposition case.

Interfaces and Interface Specification

The two aspects to be considered here in the specification of interfaces are the identification of the interfaces to be considered during the evaluation activity, and the information that needs to be provided for each interface. The proposed approach is to require the vendor to identify the set of interfaces that are associated with the security functional requirements in the PP/ST, and to map the interfaces to each SFR component. The information provided is just the identification (pointers to information in user guides, man pages, on-line documentation, or in some cases proprietary documentation of unadvertised interfaces) of the interface—no special documentation or requirements are levied on these descriptions. As these interfaces are necessarily specified in public-facing documentation, any shortcomings with respect to the evaluator's ability to test the interface could also be present for the end user, and such shortcomings would either be corrected by the vendor or noted in the evaluation report. Such shortcomings may also prove to be an area for investigation during the vulnerability analysis activity.

In addition to the identification of the interfaces, the vendor must provide a mapping of the interfaces to the applicable SFR(s). This will form the basis for the functional testing of the security mechanisms of the product.

Having the vendor provide the identification and mapping of the interfaces naturally raises the question of completeness; that is, what confidence does the end user have that all of the applicable interfaces for a given security mechanism have been identified and mapped correctly. In light of the fact that we believe that the completeness argument is very difficult to make for a modern operating system, there is no requirement for the developer to completely determine the set of external interfaces and provide that list to the evaluator, or for the evaluator to rigorously analyze the interface (and other) documentation to ensure completeness.

There are two caveats to this, however. First, since the mapped interfaces will be documented in the ST, end users reading the document will be able to tell whether a mechanism that they might be interested in has been tested. For instance, it may be the case that one method of I&A has been tested (e.g., logon using passwords), but an alternate mechanism (e.g., logon using certificates) was not. In this case, the end user would know that if they wanted to use the system in an environment that required that certificates be used to log on, additional testing (by the using organization) would be required.

Second, any interfaces not identified or mapped by the vendor are subject to investigation by the evaluation team in the vulnerability analysis activity. For instance, if the evaluation team discovers an interface not identified by the vendor that can be used to compromise the security policy (meaning the policy being enforced by the SFRs in the ST), then the vulnerability analysis can be performed and the flaw (if confirmed) would have to be addressed by the vendor.

Architectural Design Documentation

The design documentation needed to support the evaluation activities presents an architectural view of the product, focusing on a cohesive description of the security mechanisms as well as the privilege and protection mechanisms provided by the product. Initialization of the product into its initial secure state is also provided. Special attention is paid to privileged programs running in user mode or a comparable domain, since this is commonly a source of security compromises and can be analyzed more readily than kernel internals.

As this information is expected to be non-proprietary and contained within the TOE Summary Specification, another goal of this design description that the user community, including application developers and system integrators, have a clear understanding of the product's architecture, and what was included in the scope of the evaluation.

Security Functionality Description

One aspect of the design that is necessary to describe is a view of the product from a security functional perspective. This would include security functional depiction based on the SFRs at the component—or possibly a collection of components—perspective. For example, a description would present the audit subsystem , I&A subsystem, Access Control subsystem, and so on. The goal of the presentation will be to present a discussion of how the mechanisms work together, covering all of the requirements specified by the SFRs. Simply parroting back the requirements is discouraged; although on the other extreme pseudo-code should not be required either. This description will

supply system-specific detail beyond that specified in the SFRs. In some cases, an SFR maybe specified at a general or high level and it may be necessary to provide “derived” requirements (e.g., testable assertions) that reflect a specific implementation. In these cases, the developer or evaluator will map the product’s external interfaces to this granularity. This level of detailed information will aid the evaluator in their testing activities.

Privilege Architecture Description

In the Privilege Architecture Description, the vendor describes the privilege mechanism(s) that are employed, whether they are via hardware and/or software. If the product uses hardware privilege mechanisms (e.g., general-purpose operating systems, network appliances), there is a description of what hardware mechanisms are used, the level of granularity of privileges that are used (e.g., privileged/unprivileged implemented by use of ring 0 and ring 3, 4 privilege levels using four rings) what parts of the system are assigned the various privilege levels. By "parts of the system" we mean large units, such as "kernel", "device drivers", etc.; this does not require that entities operating within a single hardware privilege level be decomposed and described. The point with this portion of the description is to provide the reader a picture of the degree of separation offered by the product using the hardware.

For a product that implements software privileges, the Privilege Architecture Description explains the software privilege mechanism, as well as the level of granularity, what resources each privilege level allows access to, and what portions of the system run with each privilege. For the purposes of this discussion, "portions of the system" are grouped by hardware privilege level. For instance, if the kernel (running in a privileged mode) implements/checks 15 privileges, it is sufficient to identify the kernel as controlling those privileges without further decomposing the kernel. For portions of the system implementing privileges in the non-privileged hardware mode, each entity (usually a schedulable entity, such as a process) is identified, along with the privileges that it runs with. Note that if the product implements multiple hardware protected modes (e.g., kernel in ring 0, drivers in ring 1, and user programs in ring 3), only the hardware mode in which user-mode programs run is considered the "non-privileged hardware mode" in this discussion.

Interfaces that are available to users (administrators as well as untrusted users, either through a command-line or GUI-type interface, or a programmatic interface) that are related to the use of privilege or access to non-privileged hardware mode portions of the product (e.g., "trusted processes") are identified in the same manner as previously discussed (that is, a pointer to the description of the interface in the vendor's standard documentation is acceptable). These interfaces (as well as the software privilege mechanism) are discussed in the Administrator's Guide (some form of publically available documentation).

The privileged code and resources used by privilege code are generally held in containers that are visible to untrusted users (e.g., files). The description must contain identification of these containers, and the Administrator's Guide must contain the recommended protections for these containers (which will be verified by the evaluators). For example,

the file that contains the kernel image, executables for device drivers and trusted processes will be included in this description.

Protection Mechanism Description

In addition to using privileges, the product may employ other protection mechanisms, such as Data Execution Protection (DEP), Jails, FLASK, Address Space Layout Randomization, and stack protection mechanisms such as canaries. If the product uses such mechanisms, the description identifies how they are implemented in the product, the extent to which they are used, and any limitations that might be present (e.g., third party device drivers may not have the same requirements in this area as code written by the product developer).

Initialization Description

The startup or initialization process that the product undergoes must be documented. This explanation covers the processing that occurs at a level of detail that refers to the loading and execution of the various parts of the system, including the files identified in the Privileged Architecture Description as well as any configuration files that are used by the loaded code. The purpose here is to provide the reader an understanding of how the product reaches its secure state and is ready for interactions with untrusted entities. This description will provide enough detail that the evaluator will be able to tell if there is a way for an untrusted entity to adversely impact the initialization process.

Administrative Guidance

The architectural design will present the system's privilege mechanism that can be manipulated and in some cases must be understood at a "how to use" level by system administrators. While most of the information in the administrative guidance is well understood, it is worth pointing out that aspects of the system's privilege mechanism (presented in the architectural design) must be present in the administrative guidance and must be tested.

OSPP Functional Testing Philosophy White Paper

Functional testing is a focus area for the OSPP evaluation. The goal is provide users of the TOE confidence that functions that enforce the SFRs in the OSPP are implemented as specified through the interfaces identified in the ST. This is accomplished both through the testing called for in the assurance activities associated with each SFR, as well as testing associated with the interfaces provided by the developer (the TSFIs). This list of interfaces (contained in publically available documentation) is mapped to the SFRs; and is used to supplement the assurance activity testing effort performed during the evaluation. As a result of the testing activity, all SFRs (and test assertions derived from the SFRs) will be tested, and all TSFIs that correspond to a test assertion will be exercised.

The functional testing documentation and analysis effort comprises test coverage analysis; the developer's test evidence provided as part of the evaluation; and the evaluator's use of the developer's test evidence and the formulation of evaluation team tests. This whitepaper outlines the general approach for these activities, as well as the requirements on the contents of the test-related documentation from both the developer and evaluator.

In order to efficiently discuss what is required overall for the testing effort, the developer test evidence and developer test coverage assertion are discussed first. A discussion of the evaluator's test coverage analysis follows, which involves the use of the test evidence (as well as the documentation provided by the developer such as the Security Functionality Description, Privilege Architecture Description, etc.) produced by the developer. A discussion of the evaluation team testing effort--including the contents and level of detail that is expected in the team test plan, expectations on the evaluation team's participation in running the tests, and the results of testing--is also outlined.

Developer Test Evidence

In modern operating systems, significant effort is expended by the developer in performing functional testing. A goal in the testing effort during an evaluation activity is to use the developer test tools, artifacts, etc. to reduce the time taken to perform TOE testing without sacrificing thoroughness on the part of the evaluator. In order to achieve this objective, the evidence provided by the developer should allow the evaluator to be convinced that each test outlined in an SFR assurance activity has been addressed, and each interface that is subject to a test has been exercised with respect to its role in implementing the SFR(s) to which that interface maps.

It is important to point out that it is not required for the vendor to develop test cases specifically to address SFRs; the paradigm is that the developer provides whatever test suite they have developed – along with the mapping information outlined below—and the evaluator performs the test coverage analysis described later in this document using this information. This of course does not preclude the developer from developing such a test suite, but it is not required.

The test evidence provided by the developer includes:

- For each TSFI identified in the ST, a mapping of the applicable developer test case(s).

- For each test associated with an SFR assurance activity, the applicable developer test case(s). It's important to note that in many cases the SFR assurance activity tests are more scenario-based than interface based, which means that multiple existing developer test cases may exercise a portion of the functionality called for. While it is acceptable for the developer to specifically craft test cases that address the assurance activity tests, it is not required; it is sufficient for the developer to map the set of (existing) tests that cover the functionality to be demonstrated as specified by the assurance activity test case.
- The applicable test cases. A developer test case in the ideal sense consists of a test summary, test configuration, test instructions, test steps, and test results. The test summary contains an abstract description of the test and how the interface will be used (in a general fashion) to exercise that functionality. The test configuration consists of any test setup that must be accomplished prior to the test in order for the test to successfully run. The test instructions contain information pertaining to running the test. For automated tests, this consists of the "run this test" instruction and any information pertaining to the output. For manual tests, the test instructions are usually identical to the test steps. The test steps contain the specific steps to be followed in performing the tests. For automated tests, this consists of the contents of the test script; this could be code, scripted command line instructions, test harness language programs, etc. For manual tests, this consists of the steps that the tester must perform in order to accomplish the test (e.g., command line commands; instructions on setting radio buttons, checkboxes, and filling in parameters on GUI-based interfaces). The test results consist of the expected results from running the test. These results are the effect that the test has on the TOE, as well as how this result is checked. For automated tests this is often located within the test steps themselves, with the visible result to the user being "test passed". In other cases (most often manual tests), some observation is made while running through the test steps.

  The above description of a test case details what is commonly found from developers. Developers may have test cases that do not contain all of the information listed above, or may rely on un-written expertise of developer test personnel instead of written-down procedures. It is not required that developers provide test cases with the above sections clearly broken out or that meet any criteria; the evaluator performs a coverage analysis (described next) using the information provided by the developer and then addresses any shortfalls that are identified.
- A description of the test tools used. This description shall at a minimum allow the evaluator to determine the effect the test tool has on the TOE. Additional information in terms of instructions for using the tools, configuration of the tools, etc. may be required depending on the level of support provided by the developer in using the test tools.

Evaluator Coverage Analysis

The evaluator performs a test coverage analysis based on the information that they have available. This includes the SFR assurance activity tests and mapping to developer test cases; the interface specification; the developer-provided mapping of the interfaces to the SFRs (which is part of the delivered evidence) and test cases (discussed above); the

Security Functionality Description; the Privilege Architecture Description; and the additional information contained in the TSS. The overall goal is for the evaluator--through a combination of developer and evaluator test activities--to exercise all of the interfaces to show that the security requirements implement the SFRs.

The test coverage analysis consists of two main activities: analysis that the SFR assurance activity tests are performed; and analysis that the TSFI through which the SFR functionality can be exercised are tested. These two activities are addressed separately below.

SFR Assurance Activity Test Coverage Analysis

To perform the SFR assurance activity test coverage analysis, the evaluator performs the following steps.

1. The evaluator ensures that each SFR assurance activity test is associated with one or more developer test cases in the developer's test evidence. It should be noted that the SFR assurance activity test serves as the test objective to be used by the evaluator in their analysis (the test objective is at a similar level of detail and serves the same purpose as the developer test case summary described above).
2. In many cases, the developer's test cases will partially cover an assurance activity test, and the evaluator will need to look across several developer test cases to make a determination that the activities specified by the assurance activity test are covered. In some cases, the assurance activity test may call for parameter settings that are not present in the developer's test case. In other cases, the assurance activity may have more of a "scenario" focus, which may not be tested in the end-to-end fashion called for the assurance activity test by the developer's test cases. In those cases where the evaluator determines that the test cases provided by the developer do not address all aspects SFR assurance activity test, there are at least two courses of action. One is that the developer augments their test suite with the appropriate additional required tests. Another is that the evaluators prepare a team test to address the shortcoming. The team test case contains the information already described for a developer test case.

The information that needs to be captured in this analysis can be done in the tabular format below; this is in addition to the production of any team test cases deemed necessary.

SFR	Assurance Activity Test	Developer Test Case	Evaluation Team Test Case
SFR text	*Text from the assurance activity*	*Pointer to developer test case or cases that covers this interface/SFR*	*Pointer to evaluation team test case when developer test case is deemed inadequate, plus a rationale for what was inadequate in the developer's test(s)*

Interface-Based Test Coverage Analysis

The general approach to perform the interface-based test coverage analysis is outlined below (note that there may be some timing differences in performing these activities based on information deliveries to the evaluators and evaluator time availability; however,

all of the activities must be performed, and the order is somewhat important in order to ensure that information is used effectively).

1. The evaluator ensures that the TSFI to SFR tracing provided by the developer shows that every SFR and every TSFI is identified in the developer's test evidence, and traced to at least one test case. If an SFR is not traced to a test case, the evaluator should agree that this makes sense for that particular SFR. At this point the evaluator is not evaluating the test case; they are merely making sure that the information they will need later is available.
2. SFRs typically contain multiple testable aspects, which in this paper are identified as “test assertions.” Test assertions can be discrete testable statements taken directly from the text of the SFR or its elements; testable statements taken directly from the TSS; or testable statements that are derived as a result of information contained in the vendor-provided documentation and descriptions of the security functions as they related to the SFR(s). The evaluator expands the SFR tracings to TSFI (checked in the first step) to the test assertions that pertain to the listed SFR (see the table below). If a TSFI is mapped to an SFR, it must map to one or more of the test assertions associated with the SFR.
3. The evaluator then independently determines one or more test objectives for each test assertion mapped to a TSFI. It is important that the evaluator does this without referencing the developer test evidence in order to provide a "fresh look" at what needs to be tested at that interface. The evaluator not only considers the interface description, but also the TSS and operational guidance provided. The following are guidelines for test objective determination.
    a. For SFR elements that impose restrictions, there should be test objectives to ensure the function can succeed (when the applicable conditions are satisfied) and also to ensure that any identified restrictions are enforced.
    b. For attribute-based (e.g., privilege) restrictions, the test objectives should address cases where the minimum set of attributes allow success and where only single attributes are absent resulting in failure.
    c. For TSFIs that can operate on multiple SFR-related objects, the test objectives should address all objects.
    d. For TSFIs (generally administrative TSFIs) that involve the ability to set a security attribute, the test objectives should require verification that the setting actually has the intended effect, rather than just (for example) that the value in the GUI changed.

    The test objective should be similar in terms of level of detail to the developer test summary; it should not be detailed to the level of test steps, but it must be detailed enough so that if it were provided to two different test coders, the test cases that would be coded would test substantially the same functionality.
4. The evaluator then uses the mapping of TSFI to developer test cases to identify the test cases applicable for each TSFI. For each of the interfaces, the evaluator compares the testing performed by the developer (as captured in the developer test summary) to the test objectives developed by the evaluator. At the evaluator's

discretion, the developer test steps themselves can also be examined to provide more information as to the extent of the testing. In addition to making sure the primary security function of the interface is being tested, the evaluator also checks to ensure an appropriate "depth" of testing on the part of the developer. This involves exercising that particular interface to the extent where the evaluator has confidence that it will perform the desired function with arbitrary inputs, not just those used in the test case. For example, if an interface allows a security attribute to be set, more than one valid value, and at least one invalid value, should be contained in the test. In cases where the evaluator deems the testing effort of the developer to fall short, there are at least two courses of action. One is that the developer augments their test suite with the appropriate additional required tests. Another is that the evaluators prepare a team test to address the shortcoming. The team test contains the information already described for a developer test case.

Because of the complexity of the interfaces present in modern operating systems, it is impossible to completely test every value of every parameter of every interface. While the evaluator must run every test identified in the coverage analysis (that is, all developer tests plus any team tests) and exercise each TSFI, there is some allowance for less-than-complete testing of an interface with respect to the values that it can take on. Depth of Testing: Requirements and Documentation (below) discusses this issue—and the requirements on the evaluator activities and documentation—in more detail.

The information that needs to be captured in this analysis can be done in the tabular format below; this is in addition to the production of any team test cases deemed necessary.

SFR	**Test Assertion**	**TSFI**	**Test Objective**	**Developer Test Case**	**Evaluation Team Test Case**
SFR text	*Specific testable assertion related to the SFR (e.g., from SFR text or TSS)*	*TSFI that will be used in testing the test assertion*	*Evaluator-generated objective for testing the SFR element through the identified interface*	*Pointer to developer test case that covers this interface/SFR*	*Pointer to evaluation team test case when developer test case is deemed inadequate, plus a rationale for what was inadequate in the developer's test(s)*

Evaluator Testing Effort

In addition to the test coverage analysis outlined above, the evaluator testing effort consists of writing the evaluator test plan; potentially writing evaluation team test cases; executing the tests; and reporting the results of the tests. While the format of the documentation of the evaluator testing effort is not important, for ease of reference the following discussion uses the term "test report" to refer the collection of documentation that the evaluator uses to plan, execute, and document the results of their testing effort. This may actually consist of various documents such as a test coverage analysis, test plan, test cases; test results report, etc., but the way these documents are presented should not affect the required content. Each part of the test report is discussed in the following sections.

The test plan contains several types of information pertaining to the set-up and execution of the tests being performed by the evaluator. The evaluator is expected to run all of the tests identified in the test coverage analysis, and the test plan documents the steps by which this is to be accomplished in terms of systems and personnel needed; special test artifacts/tools to be used; and ordering of the tests.

The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. For instance, when the totality of the hardware/software interface (processor instructions, network interface, bus-accessible hardware characteristics) is identical for two hardware components (e.g., two desktop systems), then those platforms are most likely “equivalent”. However, it is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested by all tests, then no rationale is necessary. It may be the case that all platforms are included in the test plan, but only have a subset of the tests run on each of them. In this case as well rationale needs to be provided in the test plan why this is acceptable.

The test plan fully identifies the platform(s) to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools.

The evaluator shall exercise all of the test cases identified in the test coverage analysis. As mentioned previously, this does not imply that every TSFI must be tested for every possible value. However, justification needs to be provided for the tests selected; this is discussed further below in Depth of Testing: Requirements and Documentation.

The test cases contain the test procedures--both automated and manual--to be executed. It should be noted that while it's acceptable for the developer to run some portion of the tests while the evaluation team witnesses the tests, it must be clear in the test report which tests were "witnessed" by the evaluation team and which tests were run by the evaluation team. In witnessing manual tests, the evaluator shall have a step-by-step checklist to ensure that the developer is following the procedure in the test case. It is not generally acceptable for the developer to just send the evaluation team results of tests as "proof" that the tests were run.

As a result of the testing effort, the evaluator documents the actual results of the tests. This can include screen shots, logs, and checklists. This information is included in the test report. If an actual test result is different from a an expected result for a non-trivial reason (for example, the interface works differently than was thought (non-trivial) vs. a typo in the command or forgetting part of the test setup (trivial)), this is documented in the test report as well as the re-test showing the successful re-execution of the test (or if the test needed to be re-cast or removed, appropriate notation along with re-analysis of the affected interface/SFR in the test coverage analysis).

Depth of Testing: Requirements and Documentation

There are two aspects with respect to the coverage of the actual tests that deserve more detailed discussion: the testing of a security mechanism that has multiple distinct TSFI, and the amount of testing that needs to be done on an individual TSFI (which is a more general case of the first aspect).

A modern operating system contains several centralized mechanisms that are accessed in a myriad of ways; audit, access control, and identification and authentication mechanisms are just a few examples. In the documentation to be delivered by the developer, there is no longer the "traditional" TSF internals documentation that supports so-called gray-box testing approaches. While a black-box approach may have cost implications on the evaluation, it offers the highest degree of objectivity and repeatability, as well as more conclusively demonstrates that the interfaces advertised as "evaluated" work as specified. Therefore, there should be no difference in the degree of testing between different TSFIs to the same underlying mechanism.

The second aspect concerns the testing of an individual TSFI. A TSFI typical takes on several parameters, and these parameters can take on several values. Several different combinations of values can be used to demonstrate that the test assertions hold, and several other combinations of values can be used to demonstrate boundary case or parameter constraint behavior. This cumulative set of values, however, is typically a small fraction of the possible range of values and combinations of values that can be provided as test parameters for the interface. The question arises, then, with respect to when the evaluator can consider the particular TSFI "sufficiently tested".

There are no objective criteria that are possible that will hold for all TSFI in all circumstances. Further, the lack of internal documentation impacts quantitative measures such as code path coverage, issues with implementation, etc. The most important criteria, then, are that 1) all aspects of each test assertion are tested; 2) multiple (more than one) values (when applicable) are used in proving the test assertion; 3) boundary conditions are tested; and 4) the values used in performing the tests must be documented so that the actual scope of the test effort is clear to the reader.

OSPP Vulnerability Analysis White Paper

Introduction

This white paper presents a suggestion for an approach on vulnerability analysis as part of the assurance activities defined in the OSPP. The purpose of this approach is the definition of a methodology that allows the evaluator to detect critical, design-related vulnerabilities in an effective way.

The approach presented is not designed to systematically identify implementation-related vulnerabilities such as buffer overflows, non-obvious flaws in parameter validation, or non-obvious race conditions. Any approach related to the systematic identification of this type of vulnerabilities is beyond the scope of the assurance activities for an OSPP compliant operating system. During the assessment activities performed an evaluator may also detect an instance of such a vulnerability (which then of course needs to be addressed by the developer), but this happens more by chance rather than by a systematic analysis.

In order to achieve objectivity and repeatability it is important that the evaluator follows a set of well-defined activities and documents his findings such that others can follow his arguments and come to the same conclusion as the evaluator in his report. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach ensures as far as possible results that are objective and repeatable.

The expectation is that the detailed activities that take place as part of vulnerability analysis/testing will be fully developed during the course of the initial OS evaluations. It is also expected that further refinement will continue as OS evaluations continue. It is important to note that these detailed activities will be agreed upon by the OS technical community.

Vulnerability Analysis Overview

The approach of establishing general flaw hypotheses suggested here is scoped in three ways:

- A list of flaw hypotheses in the OSPP derived from Common Vulnerability Enumeration (CVE);

- A list of flaw hypotheses in the OSPP derived from lessons learned specific to that technology and other technical community input also derived from Common Weakness Enumeration (CWE) / Common Attack Pattern Enumeration and Classification (CAPEC) entries; and

- A list of flaw hypotheses derived from information available to the evaluators based on the SFRs and evidence provided by the vendor described earlier in this assurance paper, also including referenced public resources.

The first bullet list needs to be created during the course of the evaluation following the procedure described in this document. It needs to be up to date at the time of writing the Evaluation Technical Report to avoid certifying a product with publicly known vulnerabilities.

For the second bullet, although there are challenges associated with the information contained in the public sources, we believe that the list of hypotheses will be able to be refined into specific items for the evaluation team to confirm and can be included in the OSPP (as opposed to created at the time the ST is created).

However, the third type above is problematic. Because these hypotheses depend on the specific product being evaluated (and the evidence being provided as outlined in our whitepapers), we cannot list the hypotheses ahead of time; if we do, the hypotheses are so general that their utility in scoping the activity is limited.

In order to agree on the scope and high-level procedures that we think are appropriate for the vulnerability analysis/hypothesis generation activity for these types of hypotheses, the following proposals are made for what we see are acceptable principles in this area. A key aspect of this area are the cases where the evaluator needs agreement from the Scheme, and cases where the evaluator can interact directly with the developer without the need to notify the Scheme.

INFORMATION AVAILABLE TO EVALUATORS

Evaluators will naturally have access to information provided by the developer, both as outlined in our documentation whitepaper as well as information called for in assurance activities that will appear in the ST. This defines the minimum set that the developer must provide. The evaluator is free to request (without notifying the Scheme) that the developer provide the documentation necessary to meet the requirements laid out in the whitepaper/assurance activities if the evaluator feels that such documentation has not been provided. If the vendor disagrees that more information is needed, the evaluator and vendor will document their positions and obtain guidance from the Scheme on how to proceed.

FLAW GENERATION AND TESTING

The evaluators formulate flaw hypotheses based on the information provided to them as outlined in the whitepaper and assurance activities. The hypotheses are formulated during the evaluation (not prior to the evaluation) and must demonstrate that a security function of the system (as described by the SFRs in the ST) could be compromised.

If the evaluator formulates flaws that are based in material that does not need to be provided by the developer following this guide (e.g., information gleaned from an Internet mailing list, or reading interface documentation on interfaces not included in the set provided by the developer), such flaw hypotheses will be reflected with the Scheme prior to asking for developer input with respect to these flaws and after approval will be documented (source of the flaw, exploitation scenario, etc.). This can be fulfilled by providing the list for approval to the Scheme or by having meetings between CB and ITSEF, ideally also involving the developer, talking about possible vulnerabilities and

getting the Scheme approval. This way the developer can also present ideas on how to counter the hypothesis.

For either case listed above (assuming the Scheme approves further investigation of a flaw in the last case), the evaluator will refine each flaw hypothesis for the TOE and attempt to disprove it using the information provided by the developer. During this process, the evaluator is free to interact with the developer without consulting the Scheme to determine if the flaw exists, as long as there is no explicit request to the developer for additional evidence (e.g., source code, detailed design, consultation with engineering staff). The outcome of this step is a set of flaws hypotheses that have been disproved, and a set of flaw hypotheses that the evaluator feels have not been disproved.

For each flaw hypothesis that is not disproved, the evaluator will (potentially again) follow the above procedure, eventually refining his flaw description by providing an appropriate set of the following: the source documents used in formulating the hypothesis, and why it represents a potential compromise against a specific TOE function; providing the argument why the flaw hypothesis could not be proven or disproved by the evidence provided so far; and define the type of information required to investigate the flaw hypothesis further. This information is then passed to the Scheme so that the request for additional information can be approved. If approved, the developer provides the
requested evidence to disprove the flaw hypothesis (or, of course, acknowledge the flaw). The evaluator will then summarize the evidence with his judgment if the flaw hypothesis has been successfully disproved, successfully proven to have identified a flaw, or requires further investigation to be performed as part of the penetration testing effort. Again this can be dealt with in terms of meetings or written charts. It is important to have the results documented.

REPORTING

This section in its current form is explicitly intended for the piloting phase of the OSPP and to be changed afterwards.

Because the developer can provide more information than is strictly required, we feel that there must be some way to determine exactly what evidence was used and what flaws were examined. Similarly, while we understand that the amount of analysis that an evaluation facility might do in an unconstrained environment may be dependent on the Scheme and facility, we want to establish a baseline credit for vulnerability analysis activities for the OSPP and only give credit for meeting (not exceeding) that minimum bar. To help achieve this goal, we feel that the evaluation team must report all of the flaw hypotheses generated; all documentation used to generate the flaw hypotheses; and how each flaw hypothesis was resolved. In identifying the documentation used in coming up with the flaw hypotheses, the evaluation team must characterize the documentation so that a reader can determine whether it is strictly required by the whitepaper/assurance activities, and the nature of the documentation (source code, low-level design, developer engineering notebooks, etc.). At the conclusion of the evaluation, all “participating Schemes” (regarding the OSPP development project) review this information and make a determination of the impacts to supporting documents for future OSPP evaluations (for

example, if a large number of the flaw hypotheses were generated based on a certain type of documentation, then additional documentation in this area may be required for future evaluations). Therefore the participating pilot developers need to explicitly agree on the schemes sharing this information. Only agreed-upon information with respect to publically-documented vulnerabilities (the first two bullets above) will be published outside of the Scheme for reporting on the vulnerability analysis portion of the evaluation. One reason for this is the following development in the TC.

SFR-Related Assurance Activities

Introduction

This section defines SFR-related assurance activities and specifies in detail how the claims made in the SFR have to be supported by the developer documentation (TSS, potentially additional design information, interface specification, guidance, and testing) and what the evaluator needs to do to confirm that the TOE complies with the claims made.

Assurance Activities for Security Audit

Assurance Activities for FAU_GEN.1: Audit data generation

Background

Operating Systems often have extensive auditing capabilities where not all events recorded are security related. It is therefore necessary to identify the event types and related audit records the operating system is capable to record that map to the generic event types defined in FAU_GEN.1 in the Protection Profile. This is usually one or more record types in the audit trail(s) maintained by the operating system. It is the task of the evaluator to confirm that the operating system is capable to correctly generate the audit records and that the audit records contain the information required by FAU_GEN.1.

TOE Summary Specification (TSS)

Expectations

The TOE Summary Specification shall briefly describe the principle how the operating system generates audit records and name the audit mechanism used to generate the audit records required by FAU_GEN.1. Often this is a single system component and in this case it is just required to name the component and define where the component stores the audit records and how they are protected. The TSS should point to the developer documentation that defines the audit record format, either as they are stored or as they can be extracted (in the case they can only be extracted by a specific function of the TSF). It is important to describe how an administrative user (and the evaluator) can extract the audit records for further processing and analysis. The description in the TSS can be quite generic when it contains sufficient pointers to the developer documentation allowing the evaluator to generate test cases that analyze the audit records in the trail.

Evaluator activities

The evaluator analyzes the TSS and the documentation the TSS points to in order to verify that this information allows him:

1. to identify the audit trail(s) that contain the audit records related to events defined by FAU_GEN.1
2. to identify the record types for each event defined in FAU_GEN.1
3. to verify that the description of the audit record contains the information required by FAU_GEN.1

4. to identify the interface(s) that can be used to extract and analyze the audit records

Functional Specification

Expectations

Audit records are usually related to specific events that happen when the operating system is executing. Many of the events defined are directly related to user actions and in those cases the TSFI that are related to the events need to be identified. This is important to allow the evaluator to trigger specific events by using those interfaces and then verifying that the audit record expected to be generated is actually stored in the audit trail.

The evaluator therefore needs to ensure that he has obtained sufficient information to trigger the events defined in FAU_GEN.1 using the TSFI.

It is worth to note that some of the auditable events defined in FAU_GEN.1 may have several TSFI that will trigger them. In those cases assurance is needed that all of those interfaces actually also generate the related audit record. The evaluator may use design information provided by the developer that allows him to argue why there is no need to test all of the interfaces. If for example the design information clearly shows that different interfaces internally within the TSF use a common execution path and that the generation of the audit record is within this common execution path, the evaluator can justify performing tests only at one of those interfaces.

Evaluator activities

The evaluator needs to ensure that all auditable events that can be directly linked to user actions can be mapped to TSFI where the event can be triggered. The evaluator analyzes those interfaces to the extent that he does not identify obvious problems with respect to the specification of the interface, ensuring that he knows how to use the interface for testing. A more detailed analysis will be performed when the interface is used for testing.

As a result of this activity the evaluator shall for every auditable events defined in FAU_GEN.1 have a mapping to the interface(s) that can be used to trigger the event. For events where no such interface exists, the evaluator shall provide his justification why such an interface can not be expected (based on information provided by the developer) and will also indicate his view how those events may be triggered otherwise. This will be the basis for test cases that test the generation of audit records for those events.

Architectural Design

Expectations

The TOE design needs to provide an overview on the audit record generation functionality, accompanied by “assurance cases” addressing the potential problems of bypassing or otherwise disturbing audit functionality such that audit records are not generated when they should be, manipulating information to be included in audit records before and when it is collected by the audit record generation functionality, and the protection of the audit record generation functionality from being misused to generate audit records for events that did not happen. In addition the TOE design information needs to describe the format and content of the audit records required by FAU_GEN.1, mapping the details required by FAU_GEN.1 to the content of the records. The

information may (and should) be presented by references to existing developer documentation.

Evaluator activities

The TOE design information provided by the developer needs to be sufficient to address the following issues in the analysis of the functionality for FAU_GEN.1:

1. The evaluator needs to be able to identify a description of the format and structure of all the audit records that map to the auditable events required by FAU_GEN.1. The developer is free to describe the audit records as stored in the TOE internal audit trail or describe the content and format of the audit records extracted from the TOE internal audit trail by a specific tool provided by the developer as part of the TOE. The later case requires the developer to have a description of the use of this tool sufficient to extract and analyze all the audit records required by FAU_GEN.1
2. The evaluator needs to be able to identify that the audit records are actually generated by the TSF and not by a part of the TOE. The developer needs to provide sufficient arguments that the audit record generation can be influenced or even bypassed by a user.
3. The evaluator needs to be able to identify where the TSF collects the information it stores in the audit record. The developer needs to provide sufficient arguments that this information may not be subject to manipulation.
4. The evaluator needs to be able to identify that the functionality used by the TOE to generate audit records cannot be invoked by an untrusted user such that it generates an audit record for an event that never happened by using the audit functionality to produce an audit record indistinguishable from an audit record generated by the TSF for an event defined in FAU_GEN.1

User Guidance

Expectations

The user guidance related to FAU_GEN.1 needs to explain how a user authorized to extract the audit records can do this. It further needs to explain how individual information from the audit records can be presented or extracted in order to verify that all audit records expected have been generated and that the audit records contain the expected information.

Evaluator activities

The evaluator needs to ensure that the user guidance contains information about the audit records that can be generated, how to extract the audit records and how to identify the information specified in FAU_GEN.1 in the individual audit records. This information is required to be able to test FAU_GEN.1 and to ensure that all the required information is included in the different audit records that map to the requirements in FAU_GEN.1.

Testing

Expectations

The developer should be able to present test results from his test suite demonstrating that:

1. audit records have been generated when they should be and
2. audit records contain the expected information and correctly reflect the event.

Usually there is little specific testing required since the generation of audit records is (in the case of the events described in FAU_GEN.1 in the base OSPP) related to the invocation of security functions provided by the TOE that need to be tested for their specific security functionality anyhow. In order to validate the generation of the audit records, the TOE should be tested generally with all auditable events specified in FAU_GEN.1 being turned on. As long as this is not done as part of stress testing, the timing overhead associated with this extensive auditing can be neglected. Stress tests or fuzz tests that are performed in addition to pure functional testing may well be performed with a configuration where no or only a few auditable events are actually being audited.

The tests shall cover all audit events defined in FAU_GEN.1 in the Security Target to show that for each of the events defined in FAU_GEN.1.1 an audit record is created and contains the information defined for the audit records in FAU_GEN.1.2.

Evaluator activities

The evaluator analyzes the test results presented by the developer and for completeness and correctness. Note that in the case where the developer has produced a massive amount of test results resulting in a very large number of audit records being generated, the developer and the evaluator should work together on a strategy to sample those results. The sample should include cases demonstrating the correct generation of audit records for all events defined in FAU_GEN.1.1.

For those audit records not found in the sample, the evaluator defines his own test cases that are expected to cause the events related to those audit records and therefore are expected to create those records. The evaluator verifies that those audit records have been generated correctly.

After the tests have been performed, the audit records need to be extracted as part of the test results and compared to the expected audit events and content of the audit records. The evaluator needs to ensure that for each event defined in FAU_GEN.1 the expected audit records have been generated and the audit records show the expected content.

Assurance Activities for FAU_GEN.2: User Identity association

Background

In order to achieve the objective of user accountability it is required that the events recorded in the audit records can be traced to the user that caused the event, provided the event is directly related to the action of a user. This accountability has to be ensured even in cases where the subject operating on behalf of that user temporarily gets a different user ID assigned as one of its security attributes. Many operating systems allow a trusted subject’s security attribute “user ID” to be changed under the control of the OS in order to perform actions the user would not be allowed to perform using an untrusted program.

FAU_GEN.2 requires that even in those cases the identity of the user that caused the event can be associated with the user. Note that this does not require that the ID of the user that caused the event is directly placed in the audit record.

If another audit record audits this change of ID that can be easily and unambiguously linked to the audit record of the event the ability to associate such auditable events with the identity of the user that caused the event is given.

In addition an operating system may allow a user to request a service from a trusted subject using some inter-process communication function. Also in this case it must be possible to associate the identity of the user the requested the service when an audit record is generated during the processing of the request.

TOE Summary Specification (TSS)

Expectations

The TSS shall identify and describe the possible ways where an audit record is created by a subject that – at the time of the creation of the audit record – is not “bound” to the user that caused the related event.

The TSS shall explain how the identity of the user that caused the event is associated with the audit record for the event also in those cases. If the identity of that user is not part of the audit record, the TSS shall describe how someone evaluating the audit records can easily and unambiguously establish the association between the audit record and the user that caused the event recorded in the audit record.

Evaluator activities

The evaluator analyzes the TSS and the documentation the TSS points to in order to verify that this information allows him to establish an unambiguous link between the audit record and the user that caused the event.

Functional Specification

Expectations

The description of the audit records shall include all the information described as necessary to establish the association between the audit record and the user that caused the event leading to the creation of the audit record.

Evaluator activities

The evaluator verifies that the information provided allows for unambiguous association between the user that caused the event recorded in the audit record and the audit record itself.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the association between the user that caused the event and the audit record is established.

User Guidance

Expectations

If a specific configuration is required to establish the association between the user that caused the event and the audit record, it is expected that the configuration and the steps to get to this configuration are correctly and completely described in the guidance.

Evaluator activities

If a specific configuration is required to establish the association between the user that caused the event and the audit record, the evaluator follows this guidance to configure the TOE such that the association between the user that caused the event and the audit record can be established.

Testing

Expectations

The developer is expected to demonstrate in his testing that the association between the user that caused the event and the audit record can be established. Testing shall cover all cases identified in the TSS where an audit record is created by a subject that – at the time of the creation of the audit record – is not “bound” to the user that caused the related event. The test cases must identify the user(s) that caused the events.

Evaluator activities

The evaluator verifies that the test cases provided cover all cases identified in the TSS where an audit record is created by a subject that – at the time of the creation of the audit record – is not “bound” to the user that caused the related event. The evaluator extracts the audit records generated by those test cases and determines if he is able to establish the association of the event that caused the audit record to be created with the user that caused the event. The evaluator defines and executes his own test cases, collects the audit records generated and determines if he is able to establish the association of the event that caused the audit record to be created with the user that caused the event.

Assurance Activities for FAU_SAR.1: Audit review and FAU_SAR.2: Restricted audit review

Background

Reading the audit records needs to be restricted to users authorized to do so. This authorization may be assigned to a role or a privilege or there may be more complex rules governing the reading of audit data. Documentation needs to be provided that describes the interface(s) that can be used to read the audit data and the format of the audit records when read using those interfaces.

TOE Summary Specification (TSS)

Expectations

The TSS shall describe when a user is allowed to read the audit data. The TSS or documentation pointed to by the TSS need to describe the interface(s) that can be used to read the audit data and the format of the audit records when read using those interfaces.

In the case a regular file interface is used to read the audit data where the file access control functionality is used to restrict the users able to read the audit data, the format of the audit data in the file needs to be described to the extent that it is possible to correctly identify and interpret the information in the audit record.

Evaluator activities

The evaluator also analyzes the TSS and the documentation the TSS points to in order to verify that this information allows him to identify the exact conditions that need to be met for a user to be allowed to read audit data. The evaluator analyzes also the information provided on how the audit records are provided to ensure that all information required by FAU_GEN.2 is provided and that the information is suitable for the intended purpose. The intended purpose may be either reading the audit data directly (which requires them to be in printable form) or in a format suitable for post-processing by a program. In both cases the information required by FAU_GEN.2 needs to be identifiable and needs to be described such that they can be correctly interpreted.

Functional Specification

Expectations

The functional specification shall identify the interface(s) that can be used by appropriately authorized users to read the audit data. The functional specification or the guidance (or both) need to completely and correctly describe the conditions a user needs to meet in order to use those interfaces to read the audit data. The functional specification needs to describe how the audit data is presented in a way that allows extracting the information required by FAU_GEN.2 from the audit records.

Evaluator activities

The evaluator verifies that the information provided for accessing the audit data completely describe the conditions that must be met to read the audit data and that this description is consistent with the specification provided in FAU_SAR.1.1 of the Security Target. The evaluator verifies that the description how the data is provided allows him to extract the information required by FAU_GEN.1.

Note: this requirement is also satisfied if the required information is provided in the guidance documentation. In this case the evaluator uses the guidance documentation for the activities described below.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the audit data can be read and what the format of the audit data presented is.

User Guidance

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to read the audit data. The guidance needs to explain the format the audit records are presented.

Evaluator activities

See the evaluator activities for the functional specification.

Testing

Expectations

The developer is expected to demonstrate in his testing that the conditions for reading the audit data are enforced and how the audit records can be read.

Evaluator activities

The evaluator activities for this SFR consist of two main aspects:

1. Verification that only properly authorized users can access the audit data.
2. Verification that the audit data contain the required information in a form suitable for the intended processing (reading directly or post-processing by some program)

For the first aspect, the evaluator treats the conditions that must be met for reading the audit data as an access control algorithm and requires testing to be performed in the same way as outlined in the testing for discretionary access control in FDP_ACF.1.

For the second aspect the evaluator obtains audit data via the described interface(s) and verifies that the information required by FAU_GEN.1 can be extracted in the form suitable for the intended processing. The test sample needs to include audit records for all events defined in FAU_GEN.1.

Assurance Activities for FAU_SEL.1: Selective audit and FMT_MTD.1(AE): Management of TSF data: audit events

Background

For performance reasons and in order to save disk space an installation will usually not always generate audit records for all events defined in FAU_GEN.1. Therefore the OSPP requires the possibility to limit the events that are actually audited using criteria defined in FAU_SEL.1. The SFR FMT_MTD.1(AE) defines the conditions a user must satisfy in order to select the set of events that are actually audited from the overall set of auditable events defined in FAU_GEN.1.

TOE Summary Specification (TSS)

Expectations

The TSS needs to explain how the set of events that are actually audited can be limited in compliance with the criteria defined in FAU_SEL.1. The also TSS needs to describe how the management of this set of auditable events, pointing to the interface(s) used for this management.

Evaluator activities

The evaluator verifies that the explanation in the TSS and the documents pointed to by the TSS is consistent with the requirements defined in FAU_SEL.1 (i. e. allows restricting the set of audited events in accordance with the criteria defined in FAI_SEL.1) and is consistent with the conditions that must be met to perform this management operation as defined in FMT_MTD.1(AE).

Functional Specification

Expectations

The functional specification shall identify the interface(s) that can be used by appropriately authorized users to manage the set of events to be audited. The functional specification or the guidance (or both) need to completely and correctly describe the conditions a user needs to meet in order to use those interfaces to manage the event that are audited.

Evaluator activities

The evaluator verifies that the information provided for managing the events to be audited completely describe the conditions that must be met to manage the audited events and that this description is consistent with the specification provided in FAU_SEL.1 and FMT_MTD.1(AE) of the Security Target.

Note: this requirement is also satisfied if the required information is provided in the guidance documentation. In this case the evaluator uses the guidance documentation for the activities described below.

Architectural Design

Expectations

There are no further expectations on the architectural design for those SFRs than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the audit events can be managed and what the possibilities for selecting the events to be audited are.

User Guidance

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to manage the set of auditable events.

Evaluator activities

See the evaluator activities for the functional specification.

Testing

Expectations

The developer is expected to demonstrate in his testing that the conditions for managing the set of auditable events are enforced and how the auditable events can be restricted in accordance with the criteria defined in FAU_SEL.1.

Evaluator activities

The evaluator activities for those SFRs consist of three main aspects:

1. Verification that only properly authorized users can manage the set of auditable events.
2. Verification that the set of auditable events can be restricted in accordance with the criteria defined in FAU_SEL.1.
3. Verification that the TOE audits exactly the events that are defined.

For the first aspect, the evaluator treats the conditions that must be met for managing the set of auditable events as an access control algorithm and requires testing to be performed in the same way as outlined in the testing for discretionary access control in FDP_ACF.1.

For the second and third aspect the evaluator identifies test cases for each criteria mentioned in FAU_SEL.1, sets the set of auditable events in accordance with those criteria, executes a test program that would generate the appropriate audit records and verifies that the audit records are created when the criteria are defined to create them and are not created if the criteria are defined to not create the audit records.

Assurance Activities for FAU_STG.1: Protected audit trail storage

Background

Protection of the audit trail against unauthorized deletion of audit records is often achieved by using the file protection mechanism provided by the OS together with specific guidance on how to use this protection mechanism. If this is the case and no audit trail specific protection mechanisms have been implemented, the assessment of this SFR is covered by the assessment of the file protection mechanism and an assessment of the audit trail specific guidance. Only if the TOE implements audit trail specific functions for the protection of the audit records from unauthorized deletion the assurance activities for the functional specification, the architectural design, and the testing need to be performed.

TOE Summary Specification (TSS)

Expectations

The TSS shall describe how the audit records are protected from unauthorized deletion.

Evaluator activities

The evaluator analyzes the TSS and the documentation the TSS points to and identifies if the TOE uses audit trail specific protection mechanisms. If this is the case, the evaluator needs to perform the complete set of assurance activities defined for FAU_STG.1. Otherwise the evaluation only verifies that the general protection mechanisms used are covered by other SFRs (usually those for access control to storage objects) and refers to the assurance activities defined there. In this case the evaluator only verifies that the guidance provided for the protection of the audit trail ensures that the protection mechanisms are used correctly.

Functional Specification

Expectations

The functional specification shall identify the audit trail specific interface(s) used for the protection of the audit trail if such interfaces exist e. g. for managing aspects of the protection.

The functional specification needs to identify if audit trail specific interfaces for deleting audit records from the audit trail or deleting all record from the audit trail exist. If they do, the functional specification needs to describe how they can be used and how the authorization of the user of those interfaces is validated.

Evaluator activities

The evaluator verifies that the information provided for deleting records from the audit trail or all records form the audit trail completely describe the conditions that must be met to delete records from the audit trail.

Note: this requirement is also satisfied if the required information is provided in the guidance documentation. In this case the evaluator uses the guidance documentation for the activities described below.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the audit data can be read and what the format of the audit data presented is.

User Guidance

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to delete records from the audit trail.

Evaluator activities

See the evaluator activities for the functional specification.

Testing

Expectations

The developer is expected to demonstrate in his testing that the conditions for deleting records from the audit trail are enforced and that only the audit records selected are deleted.

Evaluator activities

The evaluator activities for this SFR consist of two main aspects:

1. Verification that only properly authorized users can delete records from the audit trail.
2. Verification that only the records intended to be deleted are actually deleted.

For the first aspect, the evaluator treats the conditions that must be met for deleting records from the audit trail as an access control algorithm and requires testing to be performed in the same way as outlined in the testing for discretionary access control in FDP_ACF.1.

For the second aspect the evaluator deletes selected audit records or the complete audit trail and then verifies that only those audit records have been deleted that have been selected for deletion.

Assurance Activities for FAU_STG.3: Action in case of possible audit data loss, FAU_STG.4: Prevention of audit data loss, and FMT_MTD.1(AF) Management of TSF data

Background

There may be a number of conditions that potentially could lead to a loss of audit data; reaching a defined threshold is just one of them. Another problem is a critical situation detected by the TSF that causes the TSF to shut down the TOE. In cases where the audit data is automatically transferred to another trusted IT system, any problem in the communication link with this system could potentially lead to a loss of audit data.

TOE Summary Specification (TSS)

Expectations

FAU_STG.3.1 requires the author of an ST to list in the TSS the conditions of potential loss of audit data the TSF is able to detect and describe the reaction of the TSF when such a condition is detected. This reaction may consists of a notification of some

FAU_STG.4.1 is specific for the condition that the audit trail reaches its storage limits. The TSS needs to specify the actions the TSF take when the audit trail is full, explain which audit records may get lost and which options an authorized administrator has to configure the actions taken by the TSF when the audit trail is full.

FMT_MTD.1(AF) defines the management of the actions to be taken in case of an audit storage failure.

Evaluator activities

The evaluator verifies that the explanation in the TSS and the documents pointed to by the TSS describe the reaction of the TOE to the situation described and that this is consistent with the specification in the FAU_STG.3.

The evaluator also verifies that the description of the actions taken in case the audit trail is full are consistent with the specification in FAU_STG.4.

The evaluator verifies that the TSS (and the documents pointed to by the TSS) define the possible actions taken by the TOE in case of an audit storage failure and those can be managed.

Functional Specification

Expectations

The functional specification shall identify the interface(s) that can be used by appropriately authorized users to perform potential management activities related to FAU_STG.3 and FAU_STG.4. Note that the Protection Profile does not require such management functionality to exist, but leaves the option in FAU_STG.4 to specify a function to overwrite the default values for the action to be taken when the audit trail is full.

The functional specification shall identify the interfaces that allow the management of the actions to be taken in case of an audit storage failure (which include configuration interfaces that for example allow an automatic switch to another audit storage).

Evaluator activities

The evaluator identifies from the description provided possible management actions that can be performed for FAU_STG.4. Those are mapped to the interfaces that have been identified for such management activities and analyzed for consistency with the specification in the ST.

The evaluator identifies the management interface(s) for managing the actions to be taken in case of an audit storage failure and verifies that they allow the type of management defined in FMT_MTD.1(AF) with the details mentioned in the TSS.

Architectural Design

Expectations

The architectural design needs to explain how the TSF detects that the audit storage exceeds the pre-defined limit or any other of the conditions specified in FAU_STG.3 and how the actions taken in this case are initiated by the TSF. The architectural design needs to explain how the TSF detects an audit storage failure and how it reacts to such a failure.

Evaluator activities

The evaluator checks those descriptions for consistency with the specification in the ST.

User Guidance

Expectations

If there are management activities for FAU_STG.4, the guidance needs to explain those activities, the conditions that need to be met to perform those activities and the impact of

those activities on the capability of the TOE to generate audit records. Especially if specific types of audit records get lost or if the TOE starts to overwrite old audit records, this needs to be explained in the guidance. The guidance also needs to provide advice on how to avoid getting into a situation where audit records get lost (e. g, by automatically initiating backup procedures for the audit trail). The guidance needs to explain the options an administrator has for the actions to be taken in case of an audit storage failure and what the consequences of each of those options are.

Evaluator activities

For the assessment of the management interfaces see the evaluator activities for the functional specification. The evaluator also analyzes the guidance given for preventing the loss of audit records and the management of actions in case of an audit storage failure and uses this in the development of test cases.

Testing

Expectations

The developer is expected to provide test cases and test results for the conditions defined in FAU_STG.3.1 showing that each of those conditions causes the TSF to take the actions described in FAU_STG.3.1. The developer is also expected to provide test cases showing the actions taken when the audit trail is full unless this condition cannot be reached in normal operation. In this case the developer needs to provide arguments based on the architectural design demonstrating that

a. Reaching the condition that the audit trail gets full is hard to test (even when configuring the minimum size of the audit trail allowed by the TOE)

b. If the audit trail gets full, the TSF will take the action described in FAU_STG.4.1 for this case.

The developer still has to present an estimate for the effort it would take to develop a test case for FAU_STG.4.1. The developer may choose to present a test case where the specific functionality of the TOE reacting to a full audit trail is executed in a specific environment (e. g. using a debugger or a virtualized environment) that allows to simulate the condition of a full audit trail.

Note that in the case the audit records are sent to a remote system, the situation of a full audit trail is equivalent to the situation where the remote system is no longer capable of receiving audit records. In this case the situation of a “full” audit trail can be easily simulated by disrupting the connection to the remote system.

If possible there should also be tests simulating an audit storage failure. For example in cases where audit storage is on local disks and the TOE allows for easy removing of a disk (e. g. in case of a USB disk), the developer is expected to test the case where the audit storage is on such a removable disk and this disk is removed during operation.

Evaluator activities

The evaluator verifies that all conditions listed in FAU_STG.3.1 are covered by test cases and also verifies that in each case the test results show that the actions defined in FAU_STG.3.1 have been taken.

If the developer has provided tests for FAU_STG.4.1, the evaluator will analyze the test results and determine if and why the test results show clearly that the actions specified in FAU_STG.4.1 have been taken by the TOE.

If the developer has not provided test cases for FAU_STG.4.1 with the arguments why reaching the situation where the audit trail is full is not possible without undue effort, the evaluator will provide his judgment of the arguments (including the arguments why this situation can not be tested in a specific environment) and will then analyze the arguments presented by the developer showing that the TOE will take the actions defined in FAU_STG.4.1 for the case when the audit trail is full. The evaluator will provide his judgment for those arguments in the evaluation report. The final decision if the arguments presented by the developer are acceptable is with the Certification Body.

For testing FMT_MTD.1(AF) the evaluator checks if the audit storage can be configured to be on a device that can be easily removed or can be configured to be sent to a remote system where the network connection to this system can be easily disrupted. If this is the case the evaluator tests if the correct action in case of an audit storage failure is taken by removing the disk or disconnecting the network while the TOE is operating and produces audit records.

Assurance Activities for FMT_MTD.1(AS): Management of TSF data: audit storage

Background

This function is related to the management of the audit storage, which includes a possible selection and configuration of the audit storage location and parameter, a possible creation and deletion of such storage and the clearing of the full audit trail. Note that clearing of the full audit trail is equivalent to deleting all audit records and therefore the authority to clear the audit storage as a whole must be higher than the authority required to delete individual audit records.

Note: A TOE may implement a function in the audit subsystem that allows for deleting individual audit records while the clearing of the audit storage as a whole may be implemented by file system and just require the (file system specific) authority to delete the file assigned in the configuration to be the audit trail. In this case the authorizations for the two actions are usually independent from each other and in this case the guidance needs to give advise how to co-ordinate those authorizations.

TOE Summary Specification (TSS)

Expectations

The TSS needs to describe how the storage intended to contain the audit trail is initially set up and configured, needs to describe the operations that can be performed on the audit trail storage object and how those operations are controlled.

Evaluator activities

The evaluator verifies that the description of the audit trail storage management covers the life-cycle of the audit trail storage object from its creation, assignment as the audit trail storage object and initial configuration, to its management (clearing, re-assignment

of audit trail storage (if possible), pt o deleting the audit trail storage object (if possible). For all those actions the authority required to perform the action needs to be specified. The evaluator verifies that this management model is consistent with the management of other audit trail functions.

Functional Specification

Expectations

The functional specification shall identify the interface(s) that can be used by appropriately authorized users to perform management activities related to FMT_MTD.1(AS).

Evaluator activities

The evaluator identifies from the description provided possible management actions that can be performed for FMT_MTD.1(AS). Those are mapped to the interfaces that have been identified for such management activities and analyzed for consistency with the specification in the ST.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the audit data storage object can be managed.

User Guidance

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to manage the audit trail storage object.

Evaluator activities

For the assessment of the management interfaces see the evaluator activities for the functional specification.

Testing

Expectations

The developer is expected to provide test cases and test results for the individual management activities defined in FMT_MTD.1(AS), showing that the management activities can be performed and have specified effect when the user has the required authority to perform the activity. The developer is also expected to provide test cases showing the management activities defined in FMT_MTD.1(AS) can not be performed when the user does not have the required authorization.

Evaluator activities

The evaluator verifies that all management activities listed in FMT_MTD.1(AS) are covered by test cases and also verifies that in each case the test results show that the management activity has the specified effect when the user performing the management activity is sufficiently authorized. The evaluator also verifies that the tests demonstrate that an attempt to perform a management operation specified in FMT_MTD.1(AS) without the required authorization fails.

Assurance Activities for FMT_MTD.1(AT): Management of TSF data: audit threshold

Background

This function is related to the setting of the threshold that triggers the actions defined in FAU_STG.3.1.

TOE Summary Specification (TSS)

Expectations

The TSS needs to describe how the threshold for the audit storage that triggers the actions defined in FAU_STG.3.1 can be managed.

Evaluator activities

The evaluator verifies that the functionality described in the TSS specifies how the audit trail threshold can be managed and which interface(s) can be used for this action .

Functional Specification

Expectations

The functional specification shall identify the interface(s) that can be used by appropriately authorized users to manage the audit trail threshold use by FAU_STG.3.1.

Evaluator activities

The evaluator identifies from the description how the audit trail threshold can be managed. This description needs to specify, which authorization is required to perform this management action and what the limits for the possible values of this threshold are. The evaluator verifies that the possible values for the threshold make sense (e. g. are neither negative nor larger than 100% of the audit trail capacity).

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the audit data storage object can be managed.

User Guidance

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to manage the audit trail storage threshold.

Evaluator activities

For the assessment of the management interfaces see the evaluator activities for the functional specification.

Testing

Expectations

The developer is expected to provide test cases and test results for the setting of the audit trail threshold. The developer is expected to execute the tests for FAU_STG.3.1 using different values for the audit trail threshold, showing that the actions defined in FAU_STG.3.1 are correctly taken when the threshold as defined is exceeded.

Evaluator activities

The evaluator verifies that the test results show that the actions defined in FAU_STG.3.1 are taken when the threshold is exceeded independent how the value for this threshold has been set.

Assurance Activities for User Data Protection

Assurance Activities for FDP_ACC.1 “Subset Access Control”, FDP_ACF.1 “Security attribute based access control”

Background

Operating Systems need to control access to objects they define. The OSPP base requires that an access control policy exists for all objects that allow sharing of data between different users. An operating system may implement different access control policies for different types of objects and if this is the case, the Security Target needs to have multiple instances for FDP_ACC.1 and FDP_ACF.1. The OSPP further requires that at least one access control policy for one type of named objects provides the capability to define access down to granularity of a single user.

While the OSPP requires that objects that can be used for sharing data between different users are covered by an access control policy, an operating system may use access control policies also for controlling a user’s access to specific operating system functions, use of specific privileges, or other type of “objects” not used for sharing data. A Security Target may well define also those access control policies.

The ST author needs to define in the SFRs for each access control policy:

- The types of objects, type of subjects or users and the operations covered by the access control policy
- The exact rules used by the TOE to determine if a subject/user (of the type defined in the access control policy) is allowed to perform one of the operations covered by the access control policy on an object (of the type defined in the access control policy). If the access control policy allows the definition of conflicting access rights, the algorithm needs to define how those conflicts are resolved.

Note that the same type of object may appear in different access control policies if the rules differ for different types of subjects or users or for different operations.

Note also that there may be cases where the rules used by the access control policy themselves can be managed. In this case the Security Target needs to define a fixed rule set, the guidance needs to explain how to set up this rule set for the TOE, the management of the rule set needs to be restricted to trusted administrators (or deactivated) and the administrators need to be advised in an “Evaluated Configuration Guide” to not change this rule set.

There are strong dependencies between the assessment of the access control policies themselves and the management of (user and object) security attributes as well as other TSF data used in making an access control decision. This will result in overlap in the Assurance Activities for the access control policy and the management of TSF data used in the access control policy. The evaluator should not perform assessment related to management SFRs twice but refer to the assessment performed for management SFRs in his assessment of FDP_ACC and FDP_ACF where necessary.

TOE Summary Specification (TSS)

Expectations

The TOE Summary Specification (or public documentation pointed to by the TSS) shall briefly describe the mechanisms the TOE uses to implement the access control policies and the security attributes used in the policy. For example if the TOE uses a combination of “permission bits” and “access control lists” the TOE Summary Specification needs to explain this and needs to explain how they are managed. This applies also to any other security attribute mentioned in the access control policies. Concerning the management of those security attributes, the TOE Summary Specification needs to provide information about:

- How each security attribute is initialized, especially what the default value of the security attribute is
- How the value of the security attribute can be modified (if at all) and what the rules are the TOE uses to determine if the modification is allowed

In addition the TOE Summary Specification (or public documentation pointed to by the TSS) needs to describe:

- The conditions that need to be satisfied when a user/subject requests to create a new object (for all objects mentioned in one of the access control policies),
- The rules that determine the default access rights assigned when a new object is created (for all objects mentioned in one of the access control policies),
- The conditions that need to be satisfied when a user/subject requests to delete an object (for all objects mentioned in one of the access control policies)

Evaluator Activities

The evaluator first analyzes the access control algorithm(s) defined in the SFRs (which may potentially be refined in the TSS) to validate that they are complete, providing a yes or no decision with all possible combinations of security attributes used in the rules defining the policy.

The evaluator analyzes the SFRs and the TSS for consistency. All access control policies listed in the SFRs should also be described in the TSS with the same types of objects, subjects and operations and for all security attributes mentioned in the policy the TSS needs to explain if and how they can be managed. The evaluator constructs for each access control policy a list of security attributes mentioned in the rules of the access control policy and verifies for each security attribute that the TSS either mentions it as either non-manageable (or managed internally by the TSS) or defines the rules governing the management of the security attribute. The evaluator then should have a complete model for all access control policies that define the types of subjects/users, the type of objects, and the operations covered by the access control policy as well as the full set of rules used by the TOE to determine if access is allowed by the policy. The evaluator also has the list of all security attributes used in the rules of the access control policy together with the rules that determine how those security attributes can be managed. In addition the evaluator has the rules that determine when a new object can be created together with the values of the object security attributes assigned at creation and the rules that determine when an object can be deleted.

The evaluator uses this model of the access control policies to check for completeness and for inconsistencies within this model. An example for an inconsistency would be a type of object that appears in more than one access control policy where the evaluator identifies an overlap also in the types of subjects/users and the operations and where the rules for the overlapping parts differ between the two policies. Another example of an inconsistency would be when the rules for an operation that implies another operation provide more access than the implied operation (e. g. the rules would allow a “read and write” operation in cases where it would not allow a “read” operation).

Functional Specification

Expectations

The functional specification (which is publically available) shall identify all the interfaces to the TSF where access control is enforced as well as all the interfaces used to manage the access control policy or the security attributes used in the access control policies. Each interface where access control is enforced needs to describe how the caller is informed in the case access is denied. All the interfaces need to be described such that they can be used in testing the access control policy or the management activity.

Evaluator activities

The evaluator verifies with all the interfaces identified as one where access control is enforced that the types of objects and the operations of the access control policy (or policies) addressed by the interface are identified and map to the description of the access control policy. If the description of the interface mentions more types of objects or more operations (as being subject to the access control policy) than defined in the access control policy description in the Security Target, the evaluator needs to flag this as an inconsistency. Unless the developer can provide an explanation accepted by the evaluation facility and the scheme that this is not an inconsistency, an update of the Security Target is required that removes this inconsistency.

In addition the evaluator verifies that for all security attributes that the Security Target claims are manageable, a management interface is identified in the functional specification that allows for the management action defined in the Security Target and that those interfaces are described such that they can be used for testing the management functionality.

Note: this assessment overlaps with assessment activities performed for SFRs in the management area and the evaluator ensures that the different aspects of the management activities are assessed only once. The evaluator may refer in the activities performed for FDP_ACC/FDP_ACF to the assessment performed when analyzing the SFRs related to management or vice versa.

Architectural Design

Expectations

The TOE design documentation (which consists of the TSS in the Security Target, the functional specification and any additional design related documentation provided for the evaluation) needs to explain the principles of the implementation of the access control policy (or policies), especially how and where the security attributes are stored and

maintained by the TSF. In the cases where the internal representation of those security attributes is visible at external interfaces, also the internal representation of the security attributes needs to be described in the public documentation. The TOE design needs to describe how the security attributes are protected by the mechanisms of the TOE architecture.

The TOE design documentation needs to include a justification why the access control mechanisms cannot be bypassed.

Most operating systems for access to persistent storage objects perform access control when the object is “opened” and not for each access operation to the object. As long as the user/subject is able to maintain the “open” status for the object, the access operation may be performed for the access operation checked for during “open” even if the access has been revoked afterwards. This is acceptable as long as it is described in the TOE design, functional specification, or guidance.

Sometimes a single object can be accessed using different names or links to the object. The design needs to explain that the access control rules apply regardless how the object is addressed.

Evaluator activities

The evaluator verifies that the principles of the implementation of access control policies are consistent with the description of the policies in the Security Target, the functional specification and the guidance. The evaluator verifies that that the description of the storage and management of the security attributes used in the access control policies is complete (with respect to the ones mentioned in the Security Target) and is consistent with the description in the Security Target and the guidance how they are used and managed. The best way to do this is by creating a table that maps each security attribute to its description in the Security Target, the functional specification, the design and the guidance and validating that those descriptions are consistent.

If objects can be addressed in different ways, the evaluator extracts those different ways from the TOE design, functional specification, and user guidance and determines if the TOE design provides sufficient information to ensure that regardless how the object is addressed, access control is enforced. Cases were the evaluator still is not certain may be addressed by additional test cases.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

The user guidance is expected to describe the different access control policies with their algorithms used to determine if access is allowed. The guidance also needs to explain the access control algorithm, allowing a user to understand what decision the algorithm will take based on the set of security attributes used in the rules of the algorithm.

The user guidance is expected to describe how the access control policy and the security attributes used by the access control rules can be managed, identifying the conditions that need to be satisfied to perform the individual management operations. Depending on how the developer has structured his public documentation, this information may be described

together with the interfaces used for management, which is of course an acceptable way to provide this information.

The user guidance is expected to explain how to set up the policies securely and how a user responsible for managing access control or security attributes can query the current status of security attributes that are used in the access control rules. The guidance also needs to explain the access control algorithm, allowing a user to understand what decision the algorithm will take based on the set of security attributes used in the rules of the algorithm.

There may not always be the possibility for someone allowed to manage specific security attributes to query the status of other security attributes used in an access control policy. For example a user that is allowed to modify the access control list of objects he “owns”, may not be allowed to query the list of members belonging to a group, although he is allowed to assign access rights to groups. This is not viewed as a security problem as long as this concept and how to use it securely is explained in the guidance.

Evaluator activities

The evaluator verifies that the algorithms for the different access control policies are completely and correctly described in the user guidance.

The evaluator also verifies that for all security attributes that can be managed the user guidance describes:

- How they can be managed
- The rules that define when a user is allowed to perform the individual management operations
- The effect of the management operation on the access control policy behavior
- Potential side effects that may not be immediately obvious with warnings in cases those side effects may lead to security problems. An example would be a management operation that makes the object inaccessible to any user.
- (including the conditions that need to be satisfied to perform the query operation).

In addition the evaluator verifies that the guidance describes all the steps to initialize and configure each access control policy, including the steps to set default values for security attributes, assign the required privileges to perform management operations, and activate the access control policy.

The guidance is further required to explain situations where the TOE does not implement immediate revocation of access control related security attributes, providing guidance how to avoid situations where a user may access an object for a significant amount of time after the security attributes have been modified such that his access is revoked. For example the guidance could explain how to determine the users that currently have an active access path to the object together with possible actions an authorized administrator could take to force the access path to be closed.

The evaluator will use the guidance when configuring the access control policies, defining and modifying access rights and other security attributes used in the access control algorithms when defining the test cases he needs to perform.

Testing

Expectations

Test cases may either be provided by the developer to be executed by the evaluator or be developed by the evaluator.

The test cases are required to cover:

- All access control algorithms mentioned in the Security Target
- For each access control algorithm all paths through the algorithm (as defined in the Security Target), especially each leaf in the algorithm where the algorithm terminates with a “yes” or “no” decision
- A representative set of combinations of settings of the security attributes used in the access control algorithms

Test cases need to exist also for the management functions used to manage the security attributes used in the access control algorithms. Those test cases need to cover all security attributes, each with a representative set of values for the attribute. The test cases need to show:

- That the conditions for managing the security attributes are enforced (which includes test cases where the request for management is rejected)
- That the value of the security attribute has the effect described in the access control algorithm.
- That the values of security attributes can be queried (if the necessary conditions are satisfied)

Note: those test cases will overlap with test cases required for the assessment of some management SFRs. There is of course no need to execute those tests twice, but instead the test cases may be just mapped to both the SFRs for FDP_ACC/FDP_ACF and the management SFRs.

Additional test cases are required in cases where an object can be accessed using different ways. Test cases need to exist that demonstrate that access control is enforced for each possible way to access the object. Note that not all paths through the algorithm need to be tested for each possible way to access the object.

Evaluator activity

The evaluator verifies that sufficient test cases have been provided (with their test results) showing that for each access control policy mentioned in the Security Target all paths through the access control algorithm are covered by at least one test case. He then maps the list of security attributes to test cases, showing that all security attributes are covered with a representative set of values. A representative set of values depends on the overall set of values for the security attribute and its expected effect on the access control policy.

For example an access control list is a security attribute where test cases need to exist for each possible type of access, but (of course) not for each possible user.

The evaluator also maps each management function to test cases, showing that all management functions are covered by test cases.

In most cases the developer will have significantly more test cases than required to show the coverage indicated above. When the evaluator has completed the mappings required in the description above using a subset of the test cases provided by the developer, there is no need for the evaluator to analyze the developer's test cases beyond this subset.

The evaluator will identify combinations of security attributes not found in the test cases he has analyzed and run a set of test using some of those combinations and validate that the results are consistent with the definition of the access control policy.

Assurance Activities for FDP_IFC.1 Subset information flow control and FDP_IFF.1 Simple security attributes

Background

An operating system compliant to the base OSPP is required to provide configurable functionality that allows to perform basic filtering on network traffic directed to the TOE as well as network traffic a subject generates to be sent to external IT entities. Filtering rules may be on layer 2 traffic, layer 3 traffic or both. At least the TOE needs to provide the possibility to define basic "matching" rules that allow an administrator that manages the filtering rules to prohibit traffic to and from specific unauthenticated external IT entities for layer 3 based on their IP address, TCP port number, UDP port number network protocol, and TCP header flags. For layer 2 an administrator needs to be able to define filtering rules based on MAC addresses and VLAN tags that allow or exclude traffic based on matching criteria for those attributes.

Related to those two SFRs is the SFR FMT_MSA.3(NI) which defines the conditions an administrator must meet to define the filtering rules.

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to describe the type of filtering rules for network traffic the TOE implements with:

- The network protocol(s) for which the rules apply
- The network protocol data the filtering rules can be based upon
- The criteria that can be defines for the rule to "fire"
- The possible action(s) taken when the rule "fires"

The TSS (or public documentation pointed to by the TSS) also needs to describe the management interface used to define and/or activate the filtering rules

Evaluator Activities

The evaluator verifies that the network protocols, network protocol data, the criteria for the rules to "fire" and the possible action(s) as mentioned in the TSS are consistent with

the definition in the SFRs FDP_IFC.1 and FDP_IFF.1, i. e. the criteria and rules defined in the SFRs can all be mapped to the description in the TSS or public documentation pointed to by the TSS. Note that the possibility for an administrator to define rules that match the capabilities defined in FDP_IFF.1 is verified in the assurance activities for FMT_MTD.1(NI).

Functional Specification

Expectations

The interfaces used for testing the effect of FDP_IFC.1 and FDP_IFF.1 are the external network interfaces, the interfaces a subject operating on the operating system can use to send and receive network traffic, and the interfaces an administrator can use to define and manage the filtering rules (which are analyzed and tested in the assurance activities for FMT_MTD.1(NI)). In order to verify the implementation of FDP_IFC.1 and FDP_IFF.1 the network interfaces need to be described with the specification of the network protocols they support (up to layer 3) and the interfaces a subject can use to send and receive network traffic need to be described with their parameter allowing to send and receive network data at a layer where the rules of the network information flow policy can be tested.

Evaluator activities

The evaluator verifies that the interfaces are described to the extent that he can use them to test the effect of the filtering rules.

Architectural Design

Expectations

In the case not all effects of the filtering rules can not be tested directly at the TSFI, the architectural design needs to explain which TSF internal interfaces can be used for testing the effect of the filtering rules and how those interfaces can be used for testing.

Evaluator activities

The evaluator verifies that in the case not all effects of the filtering rules can be tested at the TSFI, the sum of the TSFI described in the functional specification and the TSF internal interfaces are sufficient to test all effects of the filtering rules.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

There are no specific expectations on the user guidance.

Evaluator activities

None.

Testing

Expectations

The developer is required to present test cases that test the following cases:

- Single filter rules based on each single security attribute showing that the defined action(s) are taken in each case the rule “fires” and are not taken if the rule does not “fire”
- A combination of two or more filter rules showing that the action(s) expected to be taken for the combination of filter rules are actually taken. Note that the Assurance Activities for FMT_MSA.1(NI) require that the evaluator verifies that for each possible combination of filter rules the developer documentation allows to identify unambiguously the action(s) taken by the TOE on packets inspected. The evaluator will take this specification from the developer’s documentation to specify the expected result for the following cases:
  - A combination of filter rules that use different security attributes and define different actions
  - If possible, a combination of filter rules that use the same security attributes but define different actions
- All exceptions listed in FDP_IFF.1.4 and FDP_IFF.1.5

Evaluator activity

The evaluator shall successively configure the TOE with different filter rules in accordance with the cases defined above. The evaluator then shall initiate network traffic to the TOE from one or more external IT entities and perform tests for each set of filter rules where traffic from the external IT entity to a subject in the TOE should be blocked and where traffic from the external IT entity to a subject in the TOE should be allowed and verify that the TOE operates in accordance with its specification. Similar the evaluator shall perform tests for network traffic from a subject in the TOE to an external IT entity using different rule sets in accordance with the cases defined above and verify that the TOE operates in accordance with its specification for network traffic from a subject within the TOE to an external IT entity. The evaluator may re-use test cases provided by the developer but should use those with modified rule sets and potentially modify those test cases to cover parameter combinations not addressed in the developer’s test cases.

The test cases need to cover:

- All security attributes listed in FDP_IFF.1.3 and for each security attribute at least one test case for each possible action
- Rule sets that include multiple rules for different security attributes and test cases that test that the correct action is taken. Also in this case there needs to be at least one test case for each possible action

Assurance Activities for FDP_RIP.2 Residual information protection

Background

Residual information can potentially be present in a number of objects and resources when they are re-allocated to a different subject or user. The examples that need to be covered are:

- Residuals in persistent storage objects (file system objects) including object related TSF data (e. g. directory entries, object security attributes)
- Residuals in main memory objects
- Residuals in processor objects that can be read and written by untrusted subjects (e. g. general registers, floating point registers)

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to identify the resources that may be subject to residuals and briefly describe for each of those resources the strategy implemented by the TOE to make information stored by a subject or user unavailable before the resource is made accessible to another user or subject. If the TOE needs specific initialization and/or configuration steps to enforce object re-use for all resources, this and the steps required need to be identified in the TSS (with pointers to additional public documentation were necessary).

Evaluator Activities

The evaluator verifies that at least all resources related to objects mentioned in the access control policy SFRs (including partial release of space occupied by the object), main memory and processor resources are addressed in the description of the object re-use related description in the TSS and that the description explains sufficiently the strategy used to ensure that information about the previous content of the resource is made unavailable.

Functional Specification

Expectations

There is no direct TSF interface for object re-use. Instead the interfaces where the effect of object re-use functionality can be observed need to be identified.

Evaluator activities

For each resource identified as one that requires object re-use the evaluator identifies from the TSFI provided by the developer:

- Interfaces that can be used to release a resource
- Interfaces that can be used to re-allocate a resource
- Interfaces that can be used to read the content of a resource after re-allocation

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how object re-use is performed.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

There is no expectation on specific guidance related to object re-use unless there are management functions that can be used to specify details how object re-use is performed. If this is the case and if the TOE allows for configuration where the object re-use requirement is not satisfied, the guidance needs to describe clearly the configuration steps that have to be taken to ensure that the object re-use functionality is active.

Evaluator activities

Only in the case where the TOE needs to be specifically initialized and configured to provide object re-use capabilities for all resources the evaluator will follow the description in the TSS to identify the interfaces and parameter required to initialize and/or configure the TOE to ensure that the object re-use functionality is active. This is required before using the TOE for testing of the object re-use functionality.

Testing

Expectations

Testing is expected to cover all interfaces that can be used to allocate resources that need to be subject to object re-use and then analyze if the resource potentially contains information from its previous use by a different subject. Testing is expected to cover all attempts to obtain information left from the previous use of the resource.

Testing needs to cover at least the following cases:

- Attempts to read from persistent storage objects from areas that have not been written to since the object was created.
- Reading from main storage areas that have been obtained using dynamic storage allocation but not yet written to by the subject.
- Reading user-accessible processor register after a content switch.
- Reading from other resources listed as being subject to object re-use and allocated to the subject before information has been placed in those resources by the subject.

Evaluator activity

The evaluator verifies that all resources for which object re-use has been defined are covered by testing showing that the no access to previous information is possible. The evaluator verifies that all TSFI where newly allocated resources can be read are included in the test suite and that in no case access to the previous information is possible.

Assurance Activities for FMT_MSA.1 Management of object security attributes

Background

Object security attributes include the all object security attributes used for the enforcement of the access control policy.

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to list the object security attributes used for enforcing access control, management, and audit policies together with the rules that define when they can be managed.

Evaluator Activities

The evaluator compares the list of object security attributes mentioned in the SFRs in the rules for access control, object management and object related audit policies with the ones listed in the TSS as being manageable. For object security attributes that are mentioned in the TSS as being manageable but which are not used in any SFR, the evaluator needs to clarify their purpose. For object security attributes mentioned in SFRs but not defined as manageable in the TSS, the evaluator needs to verify that those object security attributes can not be managed by the object owner or any other user.

Functional Specification

Expectations

The interfaces used to manage object security attributes need to be identified for all object security attributes listed in the TSS as being manageable.

Evaluator activities

The evaluator verifies that for all object security attributes listed as manageable the management interfaces are identified and described and that all management actions listed in FMT_MSA.1 can be performed using those interfaces.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the object security attributes can be managed.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to manage the object security attributes.

Evaluator activities

The evaluator verifies that the conditions defined in the guidance for managing object security attributes match the conditions defined in FMT_MSA.1.

Testing

Expectations

The developer is expected to test the interfaces for the management of object security attributes as part of his functional testing. This is often done in conjunction with the testing of the SFRs where the object security attributes are used like in testing of the access control policy where those interfaces are used to set the object security attributes for testing different aspects of the access control algorithm.

Evaluator activity

The evaluator verifies that testing includes all interfaces defined for the management of object security attributes and all object security attributes, covering a sufficient set of values for the individual object security attributes. The evaluator verifies that the effect of the settings of the object security attributes are tested (often as part of the testing of the access control algorithm).

Assurance Activities for FMT_MSA.3(DAC) Static attribute initialization

Background

The default values for all security attributes used to enforce the discretionary access control policies need to be defined such that by default access is restricted to a defined set of users (usually the owner) when a new object is created. This applies to object security attributes which need to be initialized to such restrictive default values when a new object is created as well as to other security attributes used in the access control policy. Note that some object security attributes may be inherited from another object (as defined by FMT_MSA.4) and the default values in those cases are the inherited values. The rules how those inherited values are assigned are defined in FMT_MSA.4 and analyzed in the assurance activities for FMT_MSA.4.

TOE Summary Specification (TSS)

Expectations

For all object security attributes used in the discretionary access control policies the TSS (or public documentation pointed to by the TSS) needs to describe how they are initialized when a new object is created and how their initial default values are defined. The TSS also needs to describe if and how those default values can be managed, what the interfaces used for those management activities are and which conditions need to be satisfied to perform those management activities.

Evaluator Activities

The evaluator verifies that the algorithm for initializing the security attributes used in the discretionary access control policies is defined for all security attributes. The evaluator verifies that the default values restrict access to only a defined set of users (e. g. the owner and the administrators). The evaluator verifies that the default values that can be

managed and the conditions that need to be satisfied to perform those management activities are consistent with the specification in FMT_MSA.3(DAC).

Functional Specification

Expectations

The TSS identifies the interfaces that can be used to manage the default values for security attributes used to enforce the discretionary access control policies and points to the specification of those interfaces.

Evaluator activities

The evaluator verifies that the management activities mentioned in the SFR and in the TSS can be performed using those interfaces and that the description is sufficient to use those interfaces in testing.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the default values of the security attributes used to enforce the discretionary access control policies can be managed.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to manage the default values for security attributes used to enforce the discretionary access control policies.

Evaluator activities

The evaluator verifies that the conditions defined in the guidance for managing the default values of the security attributes match the conditions defined in FMT_MSA.3(DAC).

Testing

Expectations

The developer is expected to test the interfaces for the management of the default values for security attributes used to enforce the discretionary access control policies as part of his functional testing. This is often done in conjunction with the testing of the SFRs for the access control policies where those interfaces are used to set the default values for security attributes for testing different aspects of the access control algorithm.

Evaluator activity

The evaluator verifies that testing includes all interfaces defined for the management of the default values for security attributes used to enforce the discretionary access control policies, covering a sufficient set of values for the individual security attributes. The evaluator verifies that the effect of the settings of the security attributes are tested (often as part of the testing of the access control algorithms).

Assurance Activities for FMT_MSA.3(NI) Static attribute initialization

Background

The default values for all security attributes used to enforce the Network Information Flow Policy need to be defined by some set of default rules or no rules at all.

TOE Summary Specification (TSS)

Expectations

For all security attributes used in the Network Information Flow Policy the TSS (or public documentation pointed to by the TSS) needs to describe how they are initialized and how their initial default values are defined. The TSS also needs to describe if and how those default values can be managed, what the interfaces used for those management activities are and which conditions need to be satisfied to perform those management activities. Note: most likely those management actions overlap significantly with those for FMT_MTD.1(NI) and will be covered by the assurance activities defined for FMT_MTD.1(NI).

Evaluator Activities

The evaluator verifies that the algorithm for initializing the security attributes used in the Network Information Flow Policy is defined for all security attributes. The evaluator verifies that the default values satisfy the specification in FMT_MSA.3(NI). The evaluator verifies that the default values that can be managed and the conditions that need to be satisfied to perform those management activities are consistent with the specification in FMT_MSA.3(NI).

Functional Specification

Expectations

The TSS identifies the interfaces that can be used to manage the default values for security attributes used to enforce the Network Information Flow Policy and points to the specification of those interfaces.

Evaluator activities

The evaluator verifies that the management activities mentioned in the SFR and in the TSS can be performed using those interfaces and that the description is sufficient to use those interfaces in testing.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the default values of the security attributes used to enforce the Network Information Flow Policy can be managed.

User Guidance (for Administrators as well as "Regular Users")

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to manage the default values for security attributes used to enforce the Network Information Flow Policy.

Evaluator activities

The evaluator verifies that the conditions defined in the guidance for managing the default values for the security attributes match the conditions defined in FMT_MSA.3(NI).

Testing

Expectations

The developer is expected to test the interfaces for the management of the default values for security attributes used to enforce the Network Information Flow Policy as part of his functional testing. This is often done in conjunction with the testing of the SFRs for the Network Information Flow Policy where those interfaces are used to set the default values for security attributes for testing different aspects of the Network Information Flow Policy.

Evaluator activity

The evaluator verifies that testing includes all interfaces defined for the management of the default values for security attributes used to enforce the Network Information Flow Policy, covering a sufficient set of values for the individual security attributes. The evaluator verifies that the effect of the settings of the security attributes are tested (often as part of the testing of the filtering rules).

Assurance Activities for FMT_MSA.4 Security attribute value inheritance

Background

When creating a new object covered by a discretionary access control policy, the new object may inherit security attributes from an already existing object. This is often the case when objects are part of a hierarchical structure where new objects inherit security attributes from the next higher level of the hierarchy. Inheritance is not limited to hierarchical object structure but may also be the case where new objects become a member of some group and then inherit some security attributes from the group.

Inheritance is a special case for the initialization of object security attributes for new objects.

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to identify the object security attributes that are inherited when a new object is created, needs to describe what the rules for inheritance are and from where they are inherited.

Evaluator Activities

The evaluator verifies that the algorithm for inheriting security attributes used in the discretionary access control policies is defined for all security attributes that are inherited.

Functional Specification

Expectations

There are usually no interfaces related to this SFR except for the case where the inheritance rules can be managed. Inheritance is automatically performed when a new object is created. If the inheritance rules can be managed, the ST needs to define a SFR in the FMT_MTD family that describe the conditions that must be met by a user in order to be allowed to perform this management activity.

Evaluator activities

None except when the inheritance rules can be managed. In this case the interfaces for the management of the inheritance rules need to be analyzed so that they allow for the management actions defined.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the values of object security attributes are inherited.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

None.

Evaluator activities

None.

Testing

Expectations

The developer is expected to test the inheritance rules by creating new objects and then verify that the values of the object security attributes that are supposed to be inherited are correctly inherited.

Evaluator activity

The evaluator verifies that testing covers all object security attributes that can be inherited with different values for those attributes.

Assurance Activities for FMT_MTD.1(NI) Management of TSF data: network filtering rules

Background

Network data filtering rules need to be manageable, allowing a properly authorized administrator to define, query, modify, and delete the network data filtering rules. A TOE may well distinguish between the authority for the different operations, allowing for example specific users to query the filtering rules without giving them the right to modify or delete them. If such differentiations exist for different management actions, this needs to be expressed in the SFR.

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to explain how network data filtering rules can be defined and how they can be viewed, activated, modified, and deleted. The TSS also needs to identify the interfaces that can be used for those activities and the conditions a user needs to meet when performing any of those management activities.

Evaluator Activities

The evaluator verifies that the management functions and interfaces described in the TSS allow for all the management actions defined in FMT_MTD.1(NI) and that the conditions a user needs to meet to perform those activities is consistent with the description in the SFR.

Functional Specification

Expectations

The functional specification needs to define the interfaces used for the management of the network data filtering rules, defining the syntax for the management of those rules, the exact semantic of each filtering rule, the functions to define, activate, query, modify, and delete network data filtering rules. Also the conditions a user must meet to perform each of the management actions need to be defined (either in the functional specification or in the guidance documentation).

Evaluator activities

The evaluator verifies that the description of the interfaces is sufficient to perform all the management activities defined in FMT_MTD.1(NI) allowing the definition of filtering rules that cover all aspects defined in FDP_IFC.1 and FDP_IFF.1.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the network data filtering rules can be managed.

User Guidance (for Administrators as well as "Regular Users")

Expectations

Unless this is already covered in the assessment of the functional specification the guidance is expected to describe the conditions a user must satisfy to perform the different management activities for the network data filtering rules. In addition the guidance is expected to explain the semantics of the different rules and define how potential conflicts are addressed in a set of rules.

Evaluator activities

The evaluator verifies that the guidance describe the semantics of the network data filtering rules including the aspect of potentially conflicting rules in a rule set allowing the evaluator to determine for each set of rules he defines to determine the expected effect on the network traffic.

Testing

Expectations

Testing of FMT_MTD.1(NI) is expected to be performed in conjunction with the testing defined for FDP_IFC.1 and FDP_IFF.1. The management interfaces are used to define the set of network filtering rules used for the testing of the Network Information Flow Policy.

In addition the developer is expected to test that the management interfaces enforce the conditions a user must satisfy to perform the different management activities. The test cases shall cover all branches of the algorithm that determines a user's right to perform the management action, similar to testing an access control algorithm

Evaluator activity

The evaluator shall verify that the test cases cover all combinations of management activities and all branches of the algorithm that determines a user's right to perform the management action.

Assurance Activities for FMT_REV.1(OBJ) Revocation: object security attributes

Background

Revocation of object security attributes is a special case of the management of object security attributes as addressed in FMT_MSA.1. Therefore the revocation of object security attributes is handled very similar to the assessment of FMT_MSA.1 and should be performed in combination with the assurance activities for FMT_MSA.1.

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to list the object security attributes that can be revoked and needs to explain how revocation can be performed and what the conditions are that a user must satisfy to perform the revocation of an object security attribute. If those conditions are different for different objects security attributes, those differences need to be defined in the SFR.

Evaluator Activities

The evaluator compares the list of object security attributes mentioned in the SFRs with the ones listed in the TSS as being revocable and ensures that those lists are identical.

Functional Specification

Expectations

The interfaces used to revoke object security attributes need to be identified for all object security attributes listed in the TSS as being revocable.

Evaluator activities

The evaluator verifies that for all object security attributes listed as revocable the management interfaces are identified and described and that they allow for the revocation of the object security attributes.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the object security attributes can be revoked.

User Guidance (for Administrators as well as “Regular Users”)

Expectations

The guidance (or the functional specification) needs to explain the conditions that must be met to allow a user to revoke the object security attributes.

Evaluator activities

The evaluator verifies that the conditions defined in the guidance for revoking object security attributes match the conditions defined in FMT_REV.1(OBJ).

Testing

Expectations

The developer is expected to test the interfaces for the revocation of object security attributes as part of his functional testing. This is often done in conjunction with the testing of the SFRs where the object security attributes are used like in testing of the access control policy where those interfaces are used to revoke the object security attributes for testing different aspects of the access control algorithm.

Evaluator activity

The evaluator verifies that testing includes all interfaces defined for the revocation of object security attributes and all object security attributes. The evaluator verifies that the effect of the revocation of the object security attributes are tested (often as part of the testing of the access control algorithm).

Assurance Activities for Identification and Authentication

Assurance Activities for FIA_AFL.1: Authentication Failure Handling

TOE Summary Specification (TSS)

The evaluator will find the details regarding how the TOE is expected to operate when handling authentication failures in the Security Functional Description provided in the TSS. The discussion pertaining to the I&A functionality present in the TOE will describe all the methods the TOE employs to perform authentication, including what happens when a failed attempt occurs. The evaluator examines the description to ensure that each authentication method identified in this SFR is fully described and it is clear what happens when the failed attempts reach either the met or surpassed threshold. At the very least, the password-based authentication must be covered. There may be instances where the TOE behaves differently for administrators and untrusted users, and this could be either captured in the SFR by refining the requirement, iterating the requirement, or attempting to capture it in the authentication events or list of actions assignments.

Functional Specification

With an understanding of how the I&A functions are intended to operate, the evaluator turns to the interface specification to see what interfaces support I&A. The developer is required to have provided a mapping of the interfaces to the I&A functions, including those that map to FIA_UAU.5, and the evaluator ensures that the description of the methods of I&A presented in the Security Functional Description are included in the provided interfaces and vice versa. There may be interfaces for authentication that are not subject to the failure handling, and this is acceptable, as long as it is consistent with the authentication methods and authentication events listed in this SFR. For example, there may be interfaces to authenticate to the TOE that employ a smartcard, but authentication failures resulting in the use of a smartcard are not one of the authentication methods considered.

However, if during their analysis an evaluator discovers that an advertised interface whose description indicates an action will be taken relating to failed authentication attempts, and it employs the authentication method in the requirement, the evaluator must work with the developer to resolve the discrepancy. On the other hand, if the evaluator discovers an interface that employs an authentication method that is not specified in the requirement, no further action is required, since it is outside the scope of the product's claimed security functionality.

Operational User Guidance

The evaluator determines that the guidance for managing the threshold, and responding to potential actions required on their part, are consistent with the statement in the SFR.

Testing

The number of tests used to verify the TOE's behavior will, of course, depend upon the number of authentication methods that are subject to this requirement, the interfaces that invoke those methods, as well as the actions to be taken. It is suggested that the evaluator develop a matrix that contains the authentication methods to be considered, the potential authentication events that may be associated with each of the methods, and the actions that will be taken. Again, there may be various actions that are taken even given the same authentication method, and it is critical that all combinations are addressed in the testing activities. For example, when an untrusted user fails to enter the correct local password three consecutive times, their account may be disabled/locked until an administrator action is taken. On the other hand, when an administrative user fails to enter the correct local password three consecutive times their account may be disabled for 30 seconds. So the nature and number of the tests will vary due to the complexity of the TOE's failure handling mechanism.

Test 1: The evaluator, with the appropriate privilege, shall follow the operational guidance to configure the number of unsuccessful authentication attempts for each authentication method [password-based is minimally required; others may exist depending on the assignment.

Test 2: The evaluator shall attempt to authenticate successfully using the authentication method under test. After successfully authenticating, the evaluator will attempt X number of failed authentication attempts (number to be determined according to the "rules" specified in the list of authentication events. Upon satisfying the number of failed attempts, the evaluator shall observe that the TOE electrocutes the user with sufficient amperage to cause much harm.

Test 3: The evaluator shall attempt to modify the variable that enforces the limit on unsuccessful authentication attempts as an untrusted user. They shall be unsuccessful in modifying the controlling variable.

Assurance Activities for FIA_ATD.1: User Attribute Definition

TOE Summary Specification (TSS)

The evaluator will find the user security attributes maintained for each defined user enumerated in the TSS. The list of user security attributes may differ from those identified in FIA_ATD.1 and also serve to extend the minimum set in the context of

additional user security attributes assigned in FIA_ATD.1. When the list of user security attributes identified in the TSS differs from those in FIA_ATD.1, the TSS must provide a clear mapping showing the association and coverage of the required user security attributes. Any non-security related attributes associated with users need not be identified in the TSS.

Interface Specification

Given the user security attributes identified in the TSS, the evaluator turns to the interface specification to see what interfaces support FIA_ATD.1. The developer is required to have provided a mapping of the interfaces to the I&A functions, including those that map to FIA_ATD.1, and the evaluator ensures that the description of the methods available to create, view, modify, and delete the security attributes identified in the TSS are presented in the Security Functional Description.

Examples of applicable interfaces include those used to create and delete users, as well as any interfaces available to modify any of the security attributes of existing users (e.g., add/remove groups, change password).

However, if during their analysis an evaluator discovers that an advertised interface whose description indicates access to create or modify security attributes has not been mapped to FIA_ATD.1, the evaluator must work with the developer to resolve the discrepancy. On the other hand, if the evaluator discovers an interface that manipulates attributes not identified in FIA_ATD.1 (i.e., not security related), no further action is required, since it is outside the scope of the product's claimed security functionality.

Operational User Guidance

The evaluator determines that the administrative guidance for creating, viewing, modifying, and deleting user security attributes, in whole (e.g., create/delete users) or in part (e.g., change password), are consistent with FIA_ATD.1. The evaluator should, at a minimum, find instructions for creating and deleting users. Additional instructions may be available to manipulate one or more of the user security attributes individually and should be identified where available.

The description of the interface used to create or otherwise initially define users in the administrative guidance should serve to identify each of the required user attributes assignable upon creation. The description of any interface used to manipulate individual security attributes should clearly identify the applicable attribute(s).

Testing

See FMT_MTD.1(IAU) where the available interfaces are tested in conjunction with applicable restrictions.

Assurance Activities for FIA_UAU.1(RITE): Timing of Authentication

TOE Summary Specification (TSS)

The evaluators shall examine the TSS to determine that it identifies the information flows that both support remote IT authentication as well as those that might be allowed prior to the remote IT entity being authenticated. The information in the TSS pertaining to how the FDP_IFC/FDP_IFF requirements are implemented will be a key part of this

information, in that it will detail what might be allowed by the mechanism that is implemented to meet those requirements. When the TOE is configured for operational use the allowed protocols will be determined by the configuration of the parameters supported by functionality implementing FDP_IFC/FDP_IFF, so it may not be possible to provide an itemized list of what is and is not allowed prior to remote IT entity authentication. However, the evaluator shall determine that the information provided in the TSS allows a reader to understand the relationship between the configuration mechanisms supporting the FDP_IFC/FDP_IFF requirements and the resultant capabilities and functions available to remote IT entities prior to authentication.

Interface Specification

The interfaces used by remote IT entities are covered by the assurance activities for FDP_IFC.1, FDP_IFF.1, and FTP_ITC.1.

Operational User Guidance

The Operational Guidance should contain information describing the relationship between configuration the rules under which the TOE will allow information from remote IT entities and the implications of allowing flows that do not require endpoints to be authenticated. It should be possible for the administrator to determine—based on the guidance provided—what processing will be performed by the TOE (in terms of the service being allowed; for instance, allowing ICMP to pass will result in remote entities being able to “ping” the TOE without being authenticated) in response to the configuration of the rules implemented to meet the FDP_IFC/FDP_IFF requirements.

The administrative guidance will also cover the configuration of the TOE to support remote authentication; The evaluator shall examine the operational guidance to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates, etc.) to authentication are described. For each supported authentication method, the evaluator shall ensure the operational guidance provides clear instructions for successfully performing the authentication. Some of all of these configuration activities are also addressed in the assurance activity for FTP_ITC.1.

Testing

The evaluator shall perform the following test for each remote authentication method supported:

Test 1: The evaluator shall use the operational guidance to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct authentication-related information results in the ability to access the system, while providing incorrect information results in denial of access.

Tests for this capability are also addressed in the test activities for FDP_IFC.1, FDP_IFF.1, and FTP_ITC.1.

Assurance Activities for FIA_UAU.1(HU): Timing of Authentication

TOE Summary Specification (TSS)

If the TOE implements a protocol used for remote authentication of users that provides a super-set of RFC-specified functionality—or if the protocol is not specified in an RFC or

other published document—the TSS describes the portions of the protocol that are implemented that occur prior to the user being authenticated. For each action listed in the assignment that is allowed before a user logs on locally to the TOE, the TSS shall describe the functionality being provided by the TOE.

The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful authentication".

Interface Specification

The evaluator shall identify the TSFI used to authenticate to the TOE, both remotely and locally. The evaluator shall compare these interfaces to the information provided in the TSS, and determine that if the TSS describes an authorization method for a remote user (IT entity or human) or a local user, then there is an interface that corresponds to this method. If services (over and above those covered by the FDP_IFC/FDP_IFF requirements) are listed in the TSS as being available prior to user authorization, the evaluator ensures that the interfaces to these services are identified in the interface specification.

Operational User Guidance

For remote users, the operational guidance shall contain information pertaining to the configuration of the TOE to allow a user to authenticate remotely. This may involve establishing the credentials to be used by the user, as well as configuration of the TOE credentials depending on the protocol.

Testing

The evaluator shall perform the following test for each local and remote authentication method supported:

Test 1: The evaluator shall use the operational guidance to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct authentication-related information results in the ability to access the system, while providing incorrect information results in denial of access.

Test 2: For each specified service available to local users prior to authentication, the evaluation shall ensure that the service can be invoked without authentication being required.

Assurance Activities for FIA_UAU.7: Protected Authentication Feedback

TOE Summary Specification (TSS)

For each authentication method where the TOE is in control of the feedback provided to the user, the TSS indicates that the feedback provided is obscured, and how it is obscured (not provided, masked, etc.). Each authentication method must be explicitly covered, and include not only login methods, but methods that require "re-authentication" such as changing a password, for example.

Interface Specification

This information is covered by the specification of interfaces used for authentication function.

Operational User Guidance

No additional information is required specific to this functionality.

Testing

The evaluator shall perform the following test for each method of local authentication described by the TSS:

Test 1: The evaluator shall locally authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

Assurance Activities for FIA_UAU.5: Multiple authentication mechanisms

TOE Summary Specification (TSS)

The evaluator will find the available authentication mechanisms identified in the TSS. At a minimum, the TSS will describe a username/password-based authentication mechanism as well as any other mechanisms that are assigned in FIA_UAU.5.

The evaluator will also find that the username/password-based mechanism description explains the behavior of the TOE when a password is expired in a manner consistent with that selected in FIA_UAU.5.2c.

If multiple authentication mechanisms are identified, the evaluator will also find that the description explains rules associated with the additional authentication mechanisms, including rules for determining which authentication mechanism will be used in each case.

Interface Specification

Given the list of available authentication mechanisms in the TSS, the evaluator turns to the interface specification to see what interfaces support FIA_UAU.5. The developer is required to have provided a mapping of the interfaces to the I&A functions, including those that map to FIA_UAU.5, and the evaluator ensures that the description of the methods available to authenticate user identities, along with rules associated with those methods, are presented in the Security Functional Description. The descriptions should address selecting authentication methods and results of both success (e.g., create a new process) and failure (e.g., password expired) conditions.

Note that when multiple authentication methods are available, it is possible that only some of those methods are applicable to specific interfaces and that should be clearly identified.

However, if during their analysis an evaluator discovers that an advertised interface whose description indicates authentication methods that have not been mapped to FIA_UAU.5, the evaluator must work with the developer to resolve the discrepancy.

Unlike some other functions, it is generally not acceptable that available authentication methods are ignored in the context of evaluation.

Operational User Guidance

The evaluator determines that the administrative guidance for functions requiring authentication is consistent with FIA_UAU.5. The evaluator should, at a minimum, find instructions for authenticating during initial login. Additional instructions may be available for additional functions requiring authentication such as changing passwords, activating privileges, etc.

If the TOE support for multiple authentication mechanisms is configurable (e.g., to enable or set up an authentication mechanism), the guidance may also have instructions for enabling/disabling mechanisms, configuring mechanisms, defining rules for the use of mechanisms, etc. The possibilities are extensive, so the activities here may need to be augmented during an evaluation to address additional variations.

Testing

For the most part the testing activities for FIA_UAU.5 should be accomplished in conjunction with those of FIA_UAU.1. While testing for FIA_UAU.1 necessarily addresses both successful and unsuccessfully attempts to authenticate, the evaluator shall further ensure that corresponding successful and unsuccessful attempts are made in the context of each available authentication mechanism. As such, the evaluator will need to configure all possible authentication mechanisms during the course of testing to ensure that the mechanism is invoked and can result in both successful and unsuccessful cases for each applicable interface.

At a minimum, the evaluator shall also test for each interface supporting username/password-based authentication that the authentication attempt will fail when the user password is expired. Presumably, the evaluator would have already tested that authentication attempts succeed when the password is not expired per the testing described above.

Given the possibility of assigning additional authentication mechanisms, this assurance activity may need to be augmented during an evaluation to address additional possibilities.

Assurance Activities for FIA_UID.1 Timing of identification

See FIA_UAU.1

Assurance Activities for FIA_USB.1 User-subject binding

TOE Summary Specification (TSS)

The evaluator will find the user security attributes associated with each subject enumerated in the TSS. The list of user security attributes may differ from those identified in FIA_USB.1 and also serve to extend the minimum set in the context of additional user security attributes assigned in FIA_USB.1. When the list of user security attributes identified in the TSS differs from those in FIA_USB.1, the TSS must provide a clear mapping showing the association and coverage of the required user security attributes. Any non-security related attributes associated with subject need not be identified in the TSS.

While FIA_USB.1 supports assignment of user security attributes related to access control decisions, security management restrictions, and auditing, such attributes only need be identified in the SFR if they extend the minimum set of user security attributes (user identity, groups, and roles). Regardless, the evaluator will find that the TSS describes the association of all of the identified user security attributes in the context of other SFRs relate to access control, security management restrictions, and auditing. In other words, the use of each of the required user security attributes will be described in the TSS in association with at least one security function.

The Evaluator will find a description of how user security attributes are assigned to subjects. The description will describe how the user security attributes are initially assigned to a new subject, whether and how user security attributes can be changed, and how any additional security attributes might be associated with a subject. The description will serve to define all relationships between the user security attributes identified in FIA_ATD.1 and the security attributed identified in FIA_USB.1. The definition will also define all rules involved in the initial assignment and changes to security attributes associated with each subject.

If there are multiple types of subjects, potentially with different security attributes, the TSS will describe each case accordingly.

Interface Specification

Given the user security attributes identified in the TSS, the evaluator turns to the interface specification to see what interfaces support FIA_USB.1. The developer is required to have provided a mapping of the interfaces to the I&A functions, including those that map to FIA_USB.1, and the evaluator ensures that the description of the methods available to create subjects and modify security attributes associated with subjects are presented in the Security Functional Description.

Note that it is possible that interfaces may be indirect (e.g., a process created as a result of a user login) or direct (e.g., fork a new process), but they need to be identified and described in either case.

Note that it is also possible that security attributes associated with subjects cannot be changed, in which case no applicable interfaces should be identified or mapped.

The evaluator shall ensure that for each identified interface the rules for initial security attribute assignment and subsequent modifications, described in the TSS, are also described in the Security Functional Description and are consistent with the TSS.

Examples of applicable interfaces include those used to login, for a process, as well as any interfaces available to modify any of the security attributes of existing subjects (e.g., enable/disable a privilege, change real or effective user or group identifiers).

However, if during their analysis an evaluator discovers that an advertised interface whose description indicates user-subject binding functions has not been mapped to FIA_USB.1, the evaluator must work with the developer to resolve the discrepancy. On the other hand, if the evaluator discovers an interface that manipulates subject security attributes not identified in FIA_USB.1 (i.e., not security related), no further action is required, since it is outside the scope of the product’s claimed security functionality.

Operational User Guidance

The evaluator determines that the administrative guidance for creating subjects and changing security attributes associated with subjects are consistent with FIA_USB.2. The evaluator should, at a minimum, find instructions for logging in (to create a user process). Additional instructions may be available to manipulate one or more of the security attributes of subjects and should be identified where available.

The description of any interface used to manipulate security attributes of subjects should clearly identify the applicable attribute(s).

Testing

The number of tests used to verify the TOE's behavior will, of course, depend upon the number of user security attributes, the interfaces that provide access to them, and the complexity of associated restrictions. It is suggested that the evaluator develop a matrix that associates the user security attributes with interfaces available to initially assign and subsequently modify them. Note that in some cases user security attributes might be addressed collectively when an interface operates on a group of attributes simultaneously such as may be the case with functions like the UNIX 'setuid'.

The matrix should be further developed with mappings to specific rules, resulting in triples of user attribute(s), interface, and rules. Note that rules should be generally classified into two types: behavioral and restrictions. Behavioral rules serve to describe how assignment or changes occur but do not serve to limit, for example, which users or roles can perform the operation. Restrictive rules serve to describe limits for assignments and changes, such as the range of possible attributes or the roles that can make a change.

Given a list of attribute(s)/interface/rule triples, the evaluator shall perform the following tests in each case of a rule that is restrictive:

1. Perform the identified operation using instructions in the administrative guidance in order to assign or modify the identified security attribute(s) with the minimum necessary conditions to satisfy the identified rule to perform the operation. The operation should succeed. The evaluator should use an alternate interface to verify that the operation did actually succeed and the applicable security attributes have been assigned or modified. In some cases, the evaluator should be able to either refer to or build on other tests (e.g., those associated with access control, security management restrictions, or auditing) to verify the resulting security attributes have changed as expected.
2. Perform the identified operation using instructions in the administrative guidance in order to assign or modify the identified security attribute(s) with the all but the minimum necessary conditions to satisfy the identified rule to perform the operation. The operation should fail with an appropriate error. The evaluator should use an alternate interface to verify that the operation did actually not succeed and the applicable security attributes have not been assigned or modified. In some cases, the evaluator should be able to either refer to or build on other tests (e.g., those associated with access control, security management restrictions, or auditing) to verify the resulting security attributes have changed as expected.

a. This test should be repeated where multiple restrictive conditions are specified in a rule so that it is ensured that each condition is actually enforced. This is accomplished by testing with only one condition not satisfied, working through all the conditions.

Given a list of attribute(s)/interface/rule triples, the evaluator shall perform the following tests in each case of a rule that is behavioral:

1. Perform the identified operation using instructions in the administrative guidance in order to assign or modify the identified security attribute(s) in accordance with the behavioral rule. The operation should succeed. The evaluator should use an alternate interface to verify that the operation did actually succeed and the applicable security attributes have been assigned or modified. In some cases, the evaluator should be able to either refer to or build on other tests (e.g., those associated with access control, security management restrictions, or auditing) to verify the resulting security attributes have changed as expected.

    a. This test should be repeated where multiple behavioral rule components are specified in a rule so that it is ensured that each behavioral condition works as expected. This is accomplished by working through all the conditions using as few as possible in each case.

    b. Note that this test may be already addressed in the context of a test for a restive rule where one or more corresponding behavioral conditions is implied.

In general, it is expected that security attribute associated with a subject will be tested in the context of other requirements. However, the evaluator shall ensure that all security attributes are addressed in a combination of access control, security management enforcement, and audit tests. Additional tests may need to be developed in order to ensure coverage of all applicable security attributes.

Note that while the tests above should serve to verify that assignment and changes to security attributes occur as expected based on the TSS, Security Functional Description, and administrative guidance, it is not required or expected that the evaluator should comprehensively test every affected security function (access control, security management enforcement, and audit) after every possible initial or changed security attribute assignment. The basic idea is that the use of the security attributes will be tested in the context of other applicable security functions, while the focus here is on whether the assignments and changes occur correctly and only when permitted.

Assurance Activities for FIA_PK_EXT.1 Public Key and FMT_MTD.1(CM) Management of TSF data

TOE Summary Specification (TSS)

In order to show that the TSF supports the use of public keys the evaluator shall ensure that the TSS describes the following information:

If X.509v3 certificates according to RFC 5280 are used:

- For each section of RFC 5280, any statement that is not "MUST" (for example, "MAY", "SHOULD", "SHOULD NOT", etc.) shall be described so that the reader can determine whether the TOE implements that specific part of the standard;
- For each section of RFC 5280, any non-conformance to "MUST" or “SHOULD" statements shall be described;

Any TOE-specific extensions or processing that is not included in the standard that may impact the security requirements the TOE is to enforce shall be described.

If key material can be loaded and handled directly (e. g. when keys are used without digital certificates), the TSS shall describe which methods of managing the key material are supported by the TOE and what conditions need to be satisfied to perform key material management operations.

The TSS shall describe all public and private key stores implemented that contain the keys used to meet the requirements of this PP. This description shall contain information pertaining to how those keys are loaded into the store, and how the store is protected from unauthorized access. That is to say that those key material used to authorize remote IT entities can only be managed by administrative users, while untrusted users may have the ability to manage keys for their use.

Interface Specification

The collection of interfaces provided for the TOE will include those that specify how to load and manage keys / certificates. If the TOE has the capability to import certificates from a Certificate Authority (CA) or another trusted entity, the included set of interfaces will describe how the TOE can be configured to import key material / certificates from trusted authorities. This may also include how to set up a trusted channel to communicate with a CA.

Operational User Guidance

The operational guidance provides the administrator instruction as to how they configure the TOE to import key material / certificates. The importation of certificates can be from a CA, and may require the configuration steps that ensure the CA is authenticated and the communication path is protected (e.g., trusted channel). Importing other key material from a trusted entity also requires the authentication of this entity and the transfer of the key material via a trusted channel.

The guidance also instructs the administrator how they can load key material manually (e.g., through portable media), if this is supported by the TOE.

If the TOE comes preloaded with keys or certificates, the guidance instructs the administrator to manage those. This guidance will also most likely be relevant to keys or certificates that are manually loaded, or imported from a CA as well. The guidance covers how to enable or disable the trust relationship of the certificates, if this applies.

Testing

The evaluator shall devise tests that show that the TOE processes key material that conform to the implementation described in the TSS.

If certificates are used: the evaluator shall be able to form a certification path as specified in the standard and in the TSS; and are able to validate certificates as specified in the standard (certification path validation including CRL processing).

The evaluator shall perform the following tests for each function in the system that requires the use of public key encryption for authentication:

Test 1: If certificates are used: the evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

If no certificates are used: the evaluator shall demonstrate that authentication fails if no key for authentication for this user has been defined or of an incorrect key is used by the authentication partner. The evaluator shall then load a key for that user and show that authentication succeeds when using the key. The evaluator then shall delete the key for that user (or deactivate it), and show that authentication fails.

Test 2: The evaluator shall attempt to use the operational guidance to load certificates / keys from a network device and from portable/removable media (e.g., local CA, file server, USB stick, CD) that the TOE supports.

Test 3: The evaluator attempts to manage the keys/certificates that are associated with a remote IT entity that supports the functions identified in FTP_ITC.1.3. For those entities, the evaluator ensures that with administrative rights, they are able to “trust” or “untrust” those keys/certificates. Conversely, with the all but the needed rights, the evaluator attempts to modify the trust relationship; the result shall be a failed attempt.

Assurance Activities for FMT_MOF.1 Management of security functions behaviour

TOE Summary Specification (TSS)

The TSS shall describe the characteristics that are enforced for passwords, and describe the point at which the enforcement is performed.

Interface Specification

The interfaces that are used to configure the password enforcement capability are identified. The interfaces that are used to change passwords are also identified, and the evaluator ensures that these interfaces correspond to the one used for password-based authentication in the FIA_UAU requirements.

Operational User Guidance

The operational guidance shall describe the characteristics for passwords that are available; instructions for setting the enforcement mechanism; and a discussion of “strong” passwords and recommended minimum settings.

Testing

The evaluator shall also perform the following tests. Note that one or more of these tests can be performed with a single test case.

Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

Test 2: The evaluator shall set rules that require the password be composed of specific combinations of password characteristics, and then attempt to use the password. The combinations of characteristics shall cover the breadth of characteristics, but not necessarily every combination. The evaluator shall include both valid (according to the rules) and invalid (do not conform to the rules) combinations, and observe that the valid passwords are accepted and the invalid passwords are rejected. In performing this test, the evaluator shall ensure that every interface that allows passwords to be changed is exercised, but not all cases need to be run on each interface.

Test 3: The evaluator shall attempt to configure the passwords while not a member of the group that is specified as allowed to change the passwords, and observe that they are unable to configure the password rules.

Assurance Activities for FMT_MTD.1(IAT) Management of TSF data

The assurance activity for this SFR is contained within the FIA_AFL.1 requirement as they are directly related.

Assurance Activities for FMT_MTD.1(IAF) Management of TSF data

The assurance activity for this SFR is contained within the FIA_AFL.1 requirement as they are directly related.

Assurance Activities for FMT_MTD.1(IAU) Management of TSF data

TOE Summary Specification (TSS)

The relevant user security attributes have been identified in the assurance activities for FIA_ATD.1.

The evaluator shall further find that the TSS describes the restrictions that apply to creating (initializing), viewing, modifying, and deleting each of the identified user security attributes. If additional or alternate operations are available, the TSS must map them to the controlled operations identified in FMT_MTD.1(IAU). The restrictions shall identify the roles that can perform specific operations and/or rules that determine whether specific operations can be performed on each of the user security attributes. The description of restrictions must necessarily address both methods that manipulate user security attributes collectively, such as creating or deleting a user, and methods that manipulate user security attributes individual (or in groups), such as changing passwords and adding/removing group memberships or roles.

Interface Specification

The relevant interfaces have been identified in the assurance activities for FIA_ATD.1. The evaluator shall further ensure that for each identified interface the restrictions,

described in the TSS, are also described in the Security Functional Description and are consistent with the TSS.

Operational User Guidance

The relevant guidance has been identified in the assurance activities for FIA_ATD.1.

However, if the rules above are subject to change, the guidance must provide any necessary instructions. Given the open ended nature of such a possibility, this activity would need to be revisited when the rules for access to the user security attributes can be changed in the context of a given TOE.

It is not necessarily expected that the administrative guidance should identify the applicable restrictions, but if it does and the evaluator finds a contradiction between the administrative guidance and TSS or Security Functional Description, the evaluator must work with the developer to resolve the discrepancy.

Testing

The number of tests used to verify the TOE's behavior will, of course, depend upon the number of user security attributes, the interfaces that provide access to them, and the complexity of associated restrictions. It is suggested that the evaluator develop a matrix that associates the user security attributes with interfaces available to create, view, modify, or delete them. Note that in some cases user security attributes might be addressed collectively when an interface operates on a group of attributes simultaneously such as may be the case when creating or deleting a user. The matrix should be further developed with mappings to specific restrictions based on roles or rules, resulting in triples of user attribute(s), interface, and restriction.

Given a list of attribute(s)/interface/restriction triples, the evaluator shall perform the following tests in each case where the restriction is related to a role:

1. Perform the identified operation using instructions in the administrative guidance in order to manipulate the identified security attribute(s) in a role permitted to perform the operation. The operation should succeed. The evaluator should use an alternate interface to verify that the operation did actually succeed (e.g., a user was actually created or deleted).
2. Perform the identified operation using instructions in the administrative guidance in order to manipulate the identified security attribute(s) in a role not permitted to perform the operation. The operation should fail with an appropriate error. The evaluator should use an alternate interface to verify that the operation did actually not succeed (e.g., a user was not actually created or deleted).

Given a list of attribute(s)/interface/restriction triples, the evaluator shall perform the following tests in each case where the restriction is related to a rule:

1. Perform the identified operation using instructions in the administrative guidance in order to manipulate the identified security attribute(s) with the minimum necessary conditions to satisfy the identified rule to perform the

operation. The operation should succeed. The evaluator should use an alternate interface to verify that the operation did actually succeed (e.g., a user password was actually changed).

2. Perform the identified operation using instructions in the administrative guidance in order to manipulate the identified security attribute(s) with the all but the minimum necessary conditions to satisfy the identified rule to perform the operation. The operation should fail with an appropriate error. The evaluator should use an alternate interface to verify that the operation did actually not succeed (e.g., a user password was not actually changed).

    a. This test should be repeated where multiple conditions are specified in a rule so that it is ensured that each condition is actually enforced. This is accomplished by testing with only one condition not satisfied, working through all the conditions.

Assurance Activity for FPT_STM.1 and FTA_SSL

Assurance Activities for FPT_STM.1 Reliable time stamps

TOE Summary Specification (TSS)

The evaluator shall examine the TSS to ensure that it lists each function that makes use of time, including: recording of audit events, session timeout, and X.509 certificate revocation. The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions. This would include an indication of whether the function uses an internal interface to access the time or if it uses the externally visible interface.

The evaluator shall examine the TSS to gain an understanding of how system time is maintained to ensure it is reliable and monotonically increasing.

If the TOE is capable of receiving time from an external source, such as an NTP server, the TSS describes how this communication path is protected (e.g., IPsec, TLS) and ensures only authorized IT entities as defined by the administrator are able to modify the time.

Interface Specification

There should be an interface that allows all users/applications to obtain/read the system time. There will also be an interface that is used to set the local system clock. The evaluator ensures the interface specification describes how to use the interfaces to get and set time. The interface description for setting the time should specify what rights or privilege the caller must have in order to set the time.

If the TOE supports receiving time from an external entity, the interface specification describes the interface that is used to receive the time; this could be done as a manual activity, or there may be a capability that is configured that will request an update periodically.

When examining the interfaces associated with the time function, the evaluator ensures that the descriptions of the interfaces are consistent with what the TSS states about setting system time.

Operational User Guidance

The evaluator examines the operational guidance, which may reference the interface specification for the applicable interfaces, to ensure it instructs the administrator how to set the time.

If the TOE supports the use of an external entity to receive or update the time, the operational guidance provides the administrator guidance on how to setup the TOE in order to receive time from the authorized entity. The guidance should provide instructions on how to ensure the communication path is protected from attacks that could compromise the integrity of the time. For example, if the TOE is able to use an NTP server, the guidance would instruct the administrator how to configure the NTP client, and may instruct how to use a trusted channel to ensure the NTP server is

authenticated and the integrity of the information transported across the channel is either maintained, or any changes are detected.

Testing

Test 1: The evaluator uses the operational guide to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

Test2: The evaluator attempts to use the available interfaces to set the time acting as an untrusted user. The evaluator shall not be able to modify the time.

Test3: [conditional] If the TOE supports the use of an NTP server and the assignment in FTP_ITC.1.3 is used to assign NTP as a function; the evaluator shall use the operational guidance to configure the NTP client on the TOE, and set up a protected communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple cryptographic protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol.

Assurance Activities for FTA_SSL.1 TSF-initiated session locking and FTA_SSL.2 User-initiated locking

TOE Summary Specification (TSS)

The evaluator shall examine the TSS to determine how the TOE determines when the period of inactivity has been reached (e.g., no activity on the keyboard or mouse, no active programs streaming video to the monitor, no dialog boxes being popped up on the screen). The TSS also describes what controls the ability to set the time period, and whether the time period is global (i.e., system wide) or is it configurable per user account.

The evaluator also determines from the TSS description how the TOE renders the display unreadable (e.g., a user defined screen saver is activated; administrators control what is displayed when the time period is reached, a system-defined screen is presented that cannot be modified).

The evaluator shall examine the TSS to ensure it identifies what activity the system responds to (e.g., depressing key on keyboard, moving the mouse, program interacting with display) and describes how the system responds to activity and what options are presented to a user (e.g., dialog box to enter authentication credentials to unlock the session, ability to login as another user, option to shutdown the machine).

Finally, the evaluator shall examine the TSS to make certain that it describes how the user initiates a locked session, and what happens when they initiate a session-lock. It may be the case where the TSS behaves the exactly the same way as when the time out occurs. If not, the TSS describes any differences in behavior.

Interface Specification

The evaluator shall examine the interface specification for the interfaces associated with these components to determine that the capabilities present in the system defined by the TSS are consistent with what the interfaces descriptions state. At the very least, there should be interfaces that provide the ability to set the time interval, lock the session, and unlock the session.

Operational User Guidance

The evaluator shall examine the operational user guidance to ensure it instructs the administrator how to configure the inactivity time period. If the TOE provides a means to specify what is displayed when the session is locked, the operational guide describes how this is done, and the evaluator shall ensure it is consistent with the description provided in the TSS.

The evaluator shall ensure the guide describes the options that are available when the system responds to activity, and how the user can invoke those options.

The evaluator shall determine that the guide describes how users can initiate a session-lock.

Testing

The evaluator shall perform the following test:

Test 1: The evaluator follows the operational guidance to configure a few different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked after the configured time period and no remnants of data are visible on the display.

Test 2: The evaluator attempts to use the available interfaces to set the timeout period without having the proper authorizations (acting as an untrusted user). The evaluator shall fail in their attempts to modify the timeout period.

Test 3: [conditional] Variations of Test 1 and Test 2 may be necessary, depending on the complexity of the mechanism controlling the ability to set the timeout period. If the restriction is one needs to be an administrator than the test is straightforward and is as described. If there are privileges or an access control mechanism involved, the evaluator will have to determine the conditions under which to test the ability to change the timeout. In such instances, the evaluator ensures the tester has the minimum set of privileges or access control settings to change the timeout, and does so successfully. The tester than has all but one of the necessary privileges or access control settings and attempts to change the timeout, this time failing.

Test 4: The evaluator attempts to initiate the session lock capability as specified in the operational guidance. The evaluator then observes that the session is either locked after the configured time period and no remnants of data are visible on the display.

Test 5: The evaluator then ensures that re-authentication for each authentication method allowed is needed when trying to unlock the session.

Assurance Activities for Trusted Path/Channels

Assurance Activities for FTP_ITC.1 Inter-TSF trusted channel

Background

The capability to set up a trusted channel to another trusted IT product is required for an operating system compliant to the OSPP. The operating system needs to implement at

least one of the protocols SSH, TLS, or IPSec compliant with the standards referred to by the SFR implementing at least the cipher suites listed as mandatory in the SFR. Note that those mandatory cipher suites may include additional cipher suites the related RFCs define as “REQUIRED”.

TOE Summary Specification (TSS)

Expectations

The TSS (or public documentation pointed to by the TSS) needs to list the protocols specified in the SFR and the standards implemented, including options taken where the standard allows for different options.

Evaluator Activities

The evaluator verifies that the standards are referenced correctly, that they describe the protocol completely, and that any options that the standard leaves open are defined in the TSS or the developer documentation pointed to by the TSS.

Functional Specification

Expectations

For FTP_ITC.1 the interfaces are the network interfaces and the interfaces that can be used to set up a trusted channel. The interface specifications are the protocol specifications which are defined by references to the standards with a description of options taken (if the standard allows for different options). For interfaces a user can use to set up a trusted channel, the interface description needs to describe the options the user has for setting up the channel, and how to control the channel.

Note: for cases where the TSF (in accordance with the configuration defined by a trusted administrator) automatically and transparent for the user sets up a trusted channel, there may be no explicit user interface for initiating communication via a trusted channel. In this case there must be a management interface (which may be a configuration file) used by the TSF to decide when to initiate communication via a trusted channel and which options to use.

Evaluator activities

The evaluator verifies that either user interfaces exist which allows a user to initiate communication with a remote IT product using a trusted channel, or communication via a trusted channel is initiated automatically by the TSF in accordance with the administrator defined configuration. In either case the evaluator verifies that he is able to get a communication link using the trusted channel protocols specified in FTP_ITC.1 with all the options defined in the SFR. He verifies that those options can either be selected when initiating the trusted channel or can be selected with an appropriate configuration defined via a management interface.

Architectural Design

Expectations

There are no further expectations on the architectural design for this SFR than the ones defined for the TSS. The developer may well point in the TSS to existing public design documentation for further detail of this functionality.

Evaluator activities

If additional design documentation is pointed to in the TSS, the evaluator verifies that this correctly refines the statements made in the TSS and correctly describes how the object security attributes can be revoked.

User Guidance (for Administrators as well as "Regular Users")

Expectations

The guidance needs to explain how a trusted channel can be established and what the parameters for setting up a trusted channel are. The guidance needs to describe what options an administrator or a user may select and how those options affect the establishment and maintenance of the trusted channel. Especially options for selecting or excluding cipher suites that can be used as part of the protocol need to be documented, allowing an installation to restrict the cipher suites to those that are viewed as secure or are required to be used to comply with national or organizational policies.

Evaluator activities

The evaluator verifies that the guidance describes how to set up a trusted channel using the protocols defined in FTP_ITC.1 with all options defined there. Note that this activity overlaps significantly with the assessment of the functional specification and should therefore be performed together with the assessment of the interfaces.

Testing

Expectations

The developer is expected to test the protocols defined in FTP_ITC.1 with all options for the authentication of the remote IT system and all options for the cipher suites defined in FTP_ITC.1. Testing should be performed using a reference system that has a different implementation of the protocols and cipher suites to ensure that the TOE is able to set up and maintain the trusted channel to a product with an independent implementation of the protocol including the cryptographic algorithms used as part of the protocol.

Evaluator activity

The evaluator verifies that testing includes all protocols and protocol options defined. The evaluator will set up his own reference system and ensure that this system uses a different implementation of the protocols listed in FTP_ITC.1. The evaluator will perform his own tests by attempting to set up a trusted channel to an instance of the TOE. The test shall cover the following cases:

- Attempts to use options (e. g. for remote system authentication) not supported by the TOE. Those attempts need to fail.
- Attempts to use options supported by the TOE but providing incorrect authentication credentials. Those attempts need to fail.

- Attempts to use correct authentication credentials and the correct protocol version, but cipher suites not supported by the TOE. Those attempts need to fail.
- Attempts to use protocol versions not supported by the TOE (e. g. older versions of a supported protocol). Those attempts need to fail.
- Attempts to use a protocol version supported by the TOE, an authentication method supported by the TOE, correct authentication credentials, and a cipher suite supported by the TOE. Those attempts need to pass (unless there are other conditions defined in the guidance or functional specification that cause the attempt to fail in an expected way).

Mapping to the Assurance Components of the CC

Introduction

Since it is intended that the assurance activities of this Protection Profile are within the scope of the CCRA, a mapping to the assurance components listed in part 3 of the Common Criteria needs to exist and is provided in this chapter. The SFR-related assurance activities described in the previous chapters of this document are viewed as technology specific refinements of assurance components and activities defined in the CC and the CEM. As stated above it was not intended to be compliant with a specific Evaluation Assurance Level as defined in part 3 of the CC and the assurance activities have been defined without targeting such an existing Evaluation Assurance Level.

Since this document so far has focused on SFR-related assurance activities, assurance aspects not related to SFRs (like those from the ASE and ALC classes) have not been addressed so far. Components from those classes are included in the mapping described below and have to be considered in the evaluation as defined in the CC and CEM.

The chosen SAR components are a result of an analysis of either what is being covered by the additional guidance for evaluation (activities) or purely selected from the criteria as a result of discussion on assurance needs in the authoring group. The first set (namely the classes ADV, ATE and AVA) will be subject of new analysis following every refinement of the additional guidance, to put as much effort under coverage of the terms of the CCRA as possible. It has to be noted that there are activities beyond the chosen components, but higher or additional ones are not completely satisfied and therefore not part of the created assurance package.

The following sections provide the assurance components included as part of this Protection Profile. For a definition of the purpose of those components and the related evaluation assurance activities see the CC and the CEM as far specific activities have not been defined in this document.

ASE:

ASE_CCL.1, ASE_ECD.1, ASE_SPD.1 and ASE_INT.1 are required as defined in the CC and CEM.

ASE.TSS.1 is included but significant refinements have been made in the SFR-related assurance activities with respect to the information expected to be provided in the TSS.

ASE_OBJ.2 and ASE_REQ.2 will be required if additional SFR are allowed (which is not the case at least for the first trial evaluations using this Protection Profile).

ADV:

ADV_ARC.1

Note: the description provided in this document on the SFR-related assurance activities provides significant refinements with respect to the architectural design.

ADV_FSP.1

Note that in the discussion a need to go beyond ADV_FSP.1 was seen, but not a need to include all elements of ADV_FSP.2. The intention is to have all TSFI provided by the developer described to the extent that they can be used to develop test cases and correctly identify the expected result when developing test cases using the TSFI.

Note: Although no component from the ADV_TDS family is included in this mapping, design-related aspects from the description of the SFR-related assurance activities described in this document have to be considered during an evaluation. No component of the ADV_TDS family has been included since none of them fits the view on the required design evaluation aspects for products compliant with this Protection Profile.

AGD:

The components included are AGD_PRE.1 and AGD_OPE.1.

Note that the SFR-related assurance activities contain refinements for the information expected to be found in the guidance and how those aspects should be evaluated.

ALC:

The components included are: ALC_CMC.3, ALC_CMS.3, ALC_DEL.1, ALC_LCD.1 and ALC_FLR.3.

Note that ADV_DVS.1 has not been included, thus failing to satisfy the dependency from ALC_CMC.3. However, it is expected that the evaluator will examine that the CM processes described by the developer for ALC_CMC are established as described. This can be achieved for example by verifying that the described process steps are being applied during an evaluator’s on-site visit (e.g., to perform independent testing).

ATE:

The components included are: ATE_COV.2, ATE_DPT.1, ATE_FUN.1, and ATE_IND.2.

Note that refinements have been included in this document for SFR-related testing activities. It was the agreement within the group developing this Protection Profile that the evaluation needs to ensure that all SFRs have been adequately tested at the end of the evaluation. It is of secondary importance what part of the testing has been performed by the developer and what part of the testing has been performed by the evaluation facility as long as sufficient evidence exists that all SFRs have been tested at all related TSFI.

AVA:

The component included is AVA_VAN.2.