TOC

Consistency Instruction Manual

For development of

US Government Protection Profiles

For use in

Medium Robustness Environments

Information Assurance Directorate

Release 3.0

1 February 2005

Forward

(Back to TOC)

This Protection Profile (PP) Consistency Instruction Manual (CIM) for medium robustness environment was developed by the PP Review Board (PPRB) to identify and set forth a framework of consistent security requirements for the specification of products in environments requiring medium robustness, based on Version 2.2 of the CC, International Standard 15408. Details of the complete CC may be found at http://csrc.nist.gov/cc. A PP that adheres to the PP Development Process (http://niap.nist.gov/pp/pp_dev_process.pdf) and complies with the instructions of this CIM will carry the label of U.S. Government Protection Profile for a Medium Robustness Environment.

It is the intent of the PPRB that this manual be periodically updated. Feedback on its content may be forwarded to cimcomments@missi.ncsc.mil.

If you are viewing this document online, you should activate your web toolbar (View, Toolbars, Web) to maximize the use of hyperlinks embedded throughout the document.

Record of Release

(Back to TOC)

Release #	Date	Area Affected	Comment
Release 1.0	Preliminary Release September 2002	Complete Document	Preliminary Release of Consistency Guidance
Release 2.0	Initial Release March 2004	Complete Document	Initial release of the CIM
Release 3.0	1 February 2005	Various	Numerous items were changed to correct typos and to make navigation easier. Dependencies were also added to some explicit requirements.
		Introduction	Clarified the Introduction.
		Instruction #2	Clarified "Requiring Hardware and Software for medium robustness TOE" added FPT_SEP and FPT_RVM.
		Instruction #7	Clarified the definition of Assumption.
		Instruction #8	Clarified the definition of threat statements.
		Instruction #9	Added clarification of Organizational Security Policies and Security Objectives.
		Instruction #10	Text added for clarification.
		Instruction #11	Updated the International Interpretations Web Site Link
		Instruction #13	Corrected the text to reflect the "selection" convention.
		Instruction #15	Added new instruction to define "Degree of Compliance."
		Instruction # 31	Added the FPT_AMT requirement, inadvertently dropped from Version 1.
		Instruction # 36	Inserted text for explicit requirement AVA_CCA_(EXP).2
		Table 7 Policy statements	Added reference requirements to policy statements
		Forward, Glossary	Defined U.S. Government PP.
		Reference to CC version number	Replaced reference to Version 2.1 of the CC, International Standard 15408 to the updated version 2.2. Replaced NIAP interpretations with accepted international interpretations from Version 2.2; FIA_AFL.1, FIA_USB.1, FPT_RCV.2, FPT_RCV.3
		CC Abbreviations and Glossary	Added a reference of common abbreviations and terms used in the CC

Table of Contents

I. Introduction

(Back to TOC)

The PP author should use this CIM as guidance in developing any PP. This manual defines the mandatory assurance requirements that must be included within all PPs that are identified as suitable for medium robustness environments. In addition to these assurance requirements, this manual also provides minimum functionality requirements that must be addressed by the PP author, either by including each in the PP or by providing a justification for why the requirement is not applicable. This justification is not part of the PP; it is presented as a cover sheet of the PP upon submission for review by the PPRB.

The methodology and tables included within this CIM are presented as a recommended way of developing and tracking requirements throughout the development of a profile. This methodology and table structure has been used by many developers and proven beneficial and is therefore recommended. Any PP author electing to use a different methodology and requirements tracking structure should ensure that it is explained sufficiently so that the reviewers and/or users of the document can benefit from the content.

This CIM contains a template for the PP cover sheet and the table of contents (TOC); this template and TOC should be used as provided. The TOC outlines the minimum items to be included; any additional items that are needed can be added to the appropriate area in the TOC and the corresponding section of the PP.

The final authority for the technical content of the PP is the PP author; however, the PP author should ensure that the functional requirement families that are included in the PP are consistent with the technology by consulting with other experts in the technology area by direct collaboration or public vetting. The author should also review other available medium robustness PPs to maintain consistency of the components included from those families.

Improvements to this manual are being made on a regular basis and we welcome the readers' feedback. Comments on this Consistency Guide's content may be forwarded to cimcomments@missi.ncsc.mil.

II. Medium Robustness Definition

Instruction 1: Characterize Robustness Level
(Back to TOC)

All PPs should contain a discussion characterizing the level of robustness TOEs compliant with the PP can achieve, thus allowing a user of the PP to determine if a compliant TOE is appropriate for the environment in which they intend to use the TOE. The PPRB created a discussion (included below) that provides a definition of factors for TOE environments as well as an explanation of how a given level of robustness is categorized.

The intent of these new sections is to have system integrator and product vendors clearly understand the concept of robustness, what products or systems designed to meet a specific robustness level are useful for, and the suitability of a level of robustness for their application.

DODI 8500.2 February 6, 2003 says, "Robustness describes the strength of mechanism (e.g., the strength of a cryptographic algorithm) and assurance properties (i.e., confidence measures taken to ensure proper mechanism implementation) for an IA solution. The more robust a particular component is, the greater the level of confidence in the protection provided to the security services it supports. The three levels of robustness are discussed in detail in Chapter 4 in the Information Assurance Technical Framework (IATF) release 3.1 –September 2002 located at www.iatf.net It is also possible to use non-technical measures to achieve the equivalent of a level of robustness. For example, physical isolation and protection of a network can be used to provide confidentiality. In these cases, the technical solution requirement may be reduced or eliminated."

Text:
Below is text (blue text) for inclusion as Appendix D of the medium robustness PP.

General Environmental Characterization

In trying to specify the environments in which TOEs with various levels of robustness are appropriate, it is useful to first discuss the two defining factors that characterize that environment: **value of the resources** and **authorization of the entities** to those resources.

In general terms, the environment for a TOE can be characterized by the authorization (or lack of authorization) the least trustworthy entity has with respect to the highest value of TOE resources (i.e. the TOE itself and all of the data processed by the TOE).

Note that there are an infinite number of combinations of entity authorization and value of resources; this conceptually "makes sense" because there are an infinite number of potential environments, depending on how the resources are valued by the organization, and the variety of authorizations the organization defines for the associated entities. In the next section 1.2.2, these two environmental factors will be related to the robustness required for selection of an appropriate TOE.

VALUE OF RESOURCES

Value of the resources associated with the TOE includes the data being processed or used by the TOE, as well as the TOE itself (for example, a real-time control processor). "Value" is assigned by the using organization. For example, in the DoD low-value data might be equivalent to data marked "FOUO", while high-value data may be those classified Top Secret. In a commercial enterprise, low-value data might be the internal organizational structure as captured in the corporate on-line phone book, while high-value data might be corporate research results for the next generation product. Note that when considering the value of the data one must also consider the value of data or resources that are accessible through exploitation of the TOE. For example, a firewall may have "low value" data itself, but it might protect an enclave with high value data. If the firewall was being depended upon to protect the high value data, then it must be treated as a high-value-data TOE.

AUTHORIZATION OF ENTITIES

Authorization that entities (users, administrators, other IT systems) have with respect to the TOE (and thus the resources of that TOE, including the TOE itself) is an abstract concept reflecting a combination of the trustworthiness of an entity and the access and privileges granted to that entity with respect to the resources of the TOE. For instance, entities that have total authorization to all data on the TOE are at one end of this spectrum; these entities may have privileges that allow them to read, write, and modify anything on the TOE, including all TSF data. Entities at the other end of the spectrum are those that are authorized to few or no TOE resources. For example, in the case of a router, non-administrative entities may have their packets routed by the TOE, but that is the extent of their authorization to the TOE's resources. In the case of an OS, an entity may not be allowed to log on to the TOE at all (that is, they are not valid users listed in the OS's user database)**.**

It is important to note that authorization **does not** refer to the **access** that the entities actually have to the TOE or its data. For example, suppose the owner of the system determines that no one other than employees was authorized to certain data on a TOE, yet they connect the TOE to the Internet. There are millions of entities that are not **authorized** to the data (because they are not employees), but they actually have connectivity to the TOE through the Internet and thus can attempt to access the TOE and its associated resources.

Entities are characterized according to the value of resources to which they are authorized; the extent of their authorization is implicitly a measure of how trustworthy the entity is with respect to compromise of the data (that is, compromise of any of the applicable security policies; e.g., confidentiality, integrity, availability). In other words, in this model the greater the extent of an entity's authorization, the more trustworthy (with respect to applicable policies) that entity is.

SELECTION OF APPROPRIATE ROBUSTNESS LEVELS

Robustness is a characteristic of a TOE defining how well it can protect itself and its resources; a more robust TOE is better able to protect itself. This section relates the defining factors of IT environments, authorization, and value of resources to the selection of appropriate robustness levels.

When assessing any environment with respect to Information Assurance the critical point to consider is the likelihood of an attempted security policy compromise, which was characterized in the previous section in terms of entity authorization and resource value. As previously mentioned, robustness is a characteristic of a TOE that reflects the extent to which a TOE can protect itself and its resources. It follows that as the likelihood of an attempted resource compromise increases, the robustness of an appropriate TOE should also increase.

It is critical to note that several combinations of the environmental factors will result in environments in which the likelihood of an attempted security policy compromise is similar. Consider the following two cases:

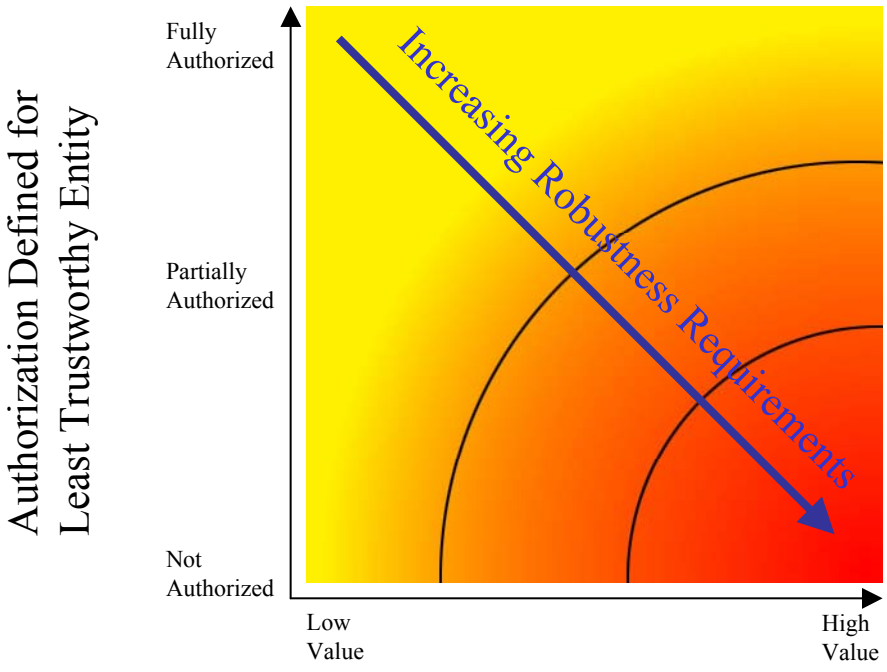
The first case is a TOE that processes only low-value data. Although the organization has stated that only its employees are authorized to log on to the system and access the data, the system is connected to the Internet to allow authorized employees to access the system from home. In this case, the least trusted entities would be unauthorized entities (e.g. non-employees) exposed to the TOE because of the Internet connectivity. However, since only low-value data are being processed, the likelihood that unauthorized entities would find it worth their while to attempt to compromise the data on the system is low and selection of a basic robustness TOE would be appropriate.

The second case is a TOE that processes high-value (e.g., classified) information. The organization requires that the TOE be stand-alone, and that every user with physical and logical access to the TOE undergo an investigation so that they are authorized to the highest value data on the TOE. Because of the extensive checks done during this investigation, the organization is assured that only highly trusted users are authorized to use the TOE. In this case, even though high value information is being processed, it is unlikely that a compromise of that data will be attempted because of the authorization and trustworthiness of the users and once again, selection of a basic robustness TOE would be appropriate.

The preceding examples demonstrated that it is possible for radically different combinations of entity authorization/resource values to result in a similar likelihood of an attempted compromise. As mentioned earlier, the robustness of a system is an indication of the protection being provided to counter compromise attempts. Therefore, a basic robustness system should be sufficient to counter compromise attempts where the likelihood of an attempted compromise is low. The following chart depicts the "universe" of environments characterized by the two factors discussed in the previous section: on one axis is the authorization defined for the least trustworthy entity, and on the other axis is the highest value of resources associated with the TOE.

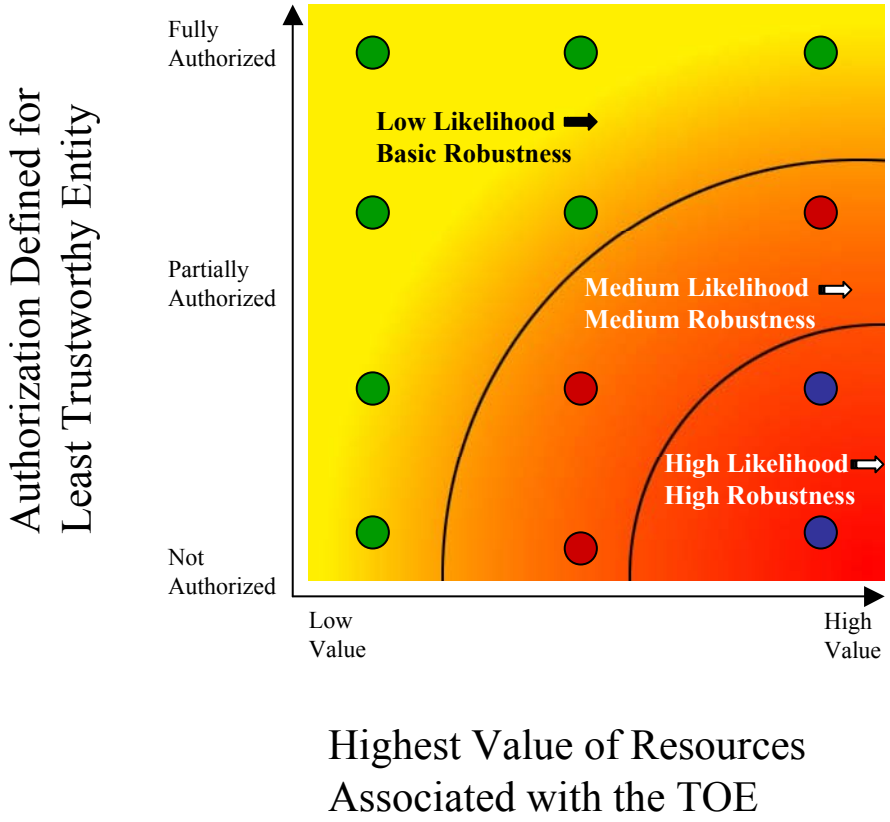
As depicted in the following figure, the robustness of the TOEs required in each environment steadily increases as one goes from the upper left of the chart to the lower right; this corresponds to the need to counter increasingly likely attack attempts by the least trustworthy entities in the environment. Note that the shading of the chart is intended to reflect- the notion that different environments engender similar levels of "likelihood of attempted compromise", signified by a similar color. Further, the delineations between such environments are not stark, but rather are finely grained and gradual.

While it would be possible to create many different "levels of robustness" at small intervals along the "Increasing Robustness Requirements" line to counter the increasing likelihood of attempted compromise due to those attacks, it would not be practical nor particularly useful. Instead, in order to implement the robustness strategy where there are only three robustness levels: Basic, Medium, and High, the graph is divided into three sections, with each section corresponding to a set of environments where the likelihood of attempted compromise is roughly similar. This is graphically depicted in the following chart.



In this second representation of environments and the robustness plane below, the "dots" represent given instantiations of environments; like-colored dots define environments with a similar likelihood of attempted compromise. Correspondingly, a TOE with a given robustness should provide sufficient protection for environments characterized by like-colored dots. In choosing the appropriateness of a given robustness level TOE PP for an environment, then, the user must first consider the lowest authorization for an entity as well as the highest value of the resources in that environment. This should result in a "point" in the chart above, corresponding to the likelihood that that entity will attempt to compromise the most valuable resource in the environment. The appropriate robustness level for the specified TOE to counter this likelihood can then be chosen.

The difficult part of this activity is differentiating the authorization of various entities, as well as determining the relative values of resources; (e.g., what constitutes "low value" data vs. "medium value" data). Because every organization will be different, a rigorous definition is not possible. In <PP Section>[1] of this PP, the targeted threat level for a medium robustness TOE is



characterized. This information is provided to help organizations using this PP - ensure that the functional requirements specified by this medium robustness PP are appropriate for their intended application of a compliant TOE.

[1] The PP author should insert the section of the PP that describes the TOE Environment.

Instruction 2: Requiring Hardware for Medium Robustness TOE
(Back to TOC)

Experience has shown that many security compromises occur when products are "composed"; that is, individual products that may be, by themselves, trustworthy, yield a vulnerable result when they are integrated together as a composite product. In order to provide the assurance necessary for products to be integrated into medium robustness environments, it is generally necessary to require that certain components of a product be evaluated as part of a TOE to give high confidence that the product is tamperproof and that the security policy is always invoked (as opposed to allowing an evaluation sponsor to remove the component from the TOE and relegate it to the environment). A particular component of note for all medium robustness products is the product's hardware. Because it is important for medium robustness products to show, through analysis and testing of an evaluation, that they are truly tamperproof and always invoke the correct policy, a medium robustness product's hardware should almost always be specified as part of the TOE that is to be compliant to a medium robustness PP. This is done through the inclusion of FPT_SEP as a requirement for the TOE. In a medium robustness TOE, this requirement cannot be met by the solely or partially by the IT Environment, and it is highly unlikely that this requirement can be met without including the underlying hardware (that supports the security functionality provided by the software components of the TOE).

It should be noted that inclusion of the hardware within the TOE boundary does not mean that the evidence about this hardware must necessarily be to the same degree of detail as the other portions of the TOE. The level of detail of design documentation and the implementation representation is dependent upon a components role in security policy enforcement (this applies to software components as well). For example, while an operating system TOE relies upon its underlying hardware for the enforcement of the TSP, the role of the hardware in this enforcement is usually only correct operation; therefore, the required details concerning the hardware would be less rigorous than the details required for the operating system. On the other hand, a network interface card (NIC) in a firewall TOE may play an important role in enforcing the firewall's information flow policy. A NIC, for performance reasons, may perform functions that impact the processing of network packets (e.g., fast FTP transfers which do not require each network packet to be processed by the TOE's network stack). There must be enough information provided for the hardware and its interaction with the TOE's software to determine the security relevance of the hardware (e.g., does it simply have to work correctly, does it have the ability to bypass policy enforcement, what is the untrusted user interface).

Medium robustness must support the argument that the TSP cannot be compromised (FPT_SEP) and that it cannot be bypassed (FPT_RVM). See Rationale for ADV_ARC_EXP.1 to help understand the use of FPT_SEP and FPT_RVM.

Required Text

Instruction 2-1: FPT_RVM.1 Reference Mediation

(Back to TOC)

FPT_RVM.1 Non-Bypassability of the TSF

FPT_RVM.1.1 The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Instruction 2-2: FPT_SEP.2 Domain Separation

(Back to TOC)

FPT_SEP.2 SFP Domain Separation

FPT_SEP.2.1 The unisolated portion of the TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.2.2 The TSF shall enforce separation between the security domains of subjects in the TSC. .

FPT_SEP.2.3 **Refinement:** The TSF shall maintain separation of the part of the TSF related to **cryptography**[2] that protects it from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to cryptography[3]

---

[2] At a minimum this separation must be maintained for the part of the TSF implementing the crypto-algorithm and the management of persistent keys.

[3] A deletion of CC text was performed in FPT_SEP.2.3. Rationale: The words “security domain” were deleted to scope the requirement an address space. The words "their”, “them", and “those SPFs” were deleted for grammatical reasons since this element refers to cryptography and not SPFs.

FPT_SEP.2.3 **Refinement:** The TSF shall maintain **separation of** the part of the TSF related to **cryptography** in ~~a security domain for their own execution~~ that protects ~~them~~ **it** from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to ~~those SPFs~~ **cryptography**.

Instruction 3: Uses of Medium Robustness
(Back to TOC)

The PPRB recognized the importance of a clear understanding of the TSF specified in terms of applicable assumptions, threats and policies which are related to or appropriate for a particular robustness levels.

Therefore, it is required that PP authors include in section 3 of all PPs a discussion relating the specified TOE robustness level to the formation of applicable assumptions, threats and policies of the TOE security environment (TSE).

Text for medium robustness PPs:

A medium robustness TOE is considered sufficient protection for environments where the likelihood of an attempted compromise is medium. This implies that the motivation of the threat agents will be average in environments that are suitable for TOEs of medium robustness. Note that while highly sophisticated threat agents will not be motivated to use great expertise or extensive resources in an environment where medium robustness is suitable, the wide spread availability of exploits and hacking tools available on the Internet provide less sophisticated threat agents with expertise (and indirectly resources) that they otherwise might not have access to.

The medium motivation of the threat agents can be reflected in a variety of ways. One possibility is that the value of the data processed or protected by the TOE will be only medium, thus providing little motivation of even a totally unauthorized entity to attempt to compromise the data. Another possibility, (where higher value data is processed or protected by the TOE) is that the procuring organization will provide environmental controls (that is, controls that the TOE itself does not enforce) in order to ensure that threat agents that have generally high motivation levels (because of the value of the data) cannot logically or physically access the TOE (e.g., all users are “vetted” to help ensure their trustworthiness, and connectivity to the TOE is restricted).

Instruction 4: Assurance Requirements for Medium Robustness
(Back to TOC)

A TOE that has been evaluated against the requirements of a medium robustness PP has several differences from one that has been evaluated against a Basic Robustness PP. The following list is some areas where Medium and Basic Robustness profiles differ:

- Roles and remote administration (FMT_SMR)
- Hardware is included in the TOE
- Toe access requirements (FTA requirements)
- Potential violation analysis (FAU_SAA requirements)
- Assurance requirements

The Security Assurance Requirements drawn or derived from the CC for Information Technology Security Evaluation, Part 3, dated January 2004, Version 2.2 of CCIMB-2004-01-002 which collectively define "medium robustness" include the following:

The assurance requirements were originally based upon Evaluated Assurance Level (EAL) 4. In order to gain the necessary level of assurance for medium robustness environments explicit requirements have been created for some families in the ADV class both to remove ambiguity in the existing ADV requirements as well as to provide greater assurance than that associated with EAL4. The set of assurance components are noted in the following table. Those labeled with an EXP suffix are further described in various instructions in this document. Requirements bolded that are not explicit requirements are those that have been selected to augment the CC EAL4 for medium robustness PPs.

Assurance Class	Assurance Components	Assurance Components Description
Configuration Management	ACM_AUT.1	Partial CM automation
	ACM_CAP.4	Generation support and acceptance procedures
	ACM_SCP.2	Problem tracking CM coverage
Delivery and Operation	ADO_DEL.2	Detection of modification
	ADO_IGS.1	Installation, generation, and start-up procedures
Development	**ADV_FSP_(EXP).1**	**Functional specification With Complete Summary, see Instruction 33:3**
	ADV_HLD_(EXP).1	**Security-enforcing high-level design, see Instruction 33:4**
	ADV_ARC_(EXP).1	**Architectural Design with Justification, see Instruction 33:1**
	ADV_INT_(EXP).1	**Modular decomposition, see Instruction 33:2**

Assurance Class	Assurance Components	Assurance Components Description
	ADV_IMP.2	**Implementation of the TSF**
	ADV_LLD_(EXP).1	**Security-enforcing low-level design, see Instruction 33:5**
	ADV_RCR.1	Informal correspondence demonstration
	ADV_SPM.1	Informal TOE security policy model
Guidance Documents	AGD_ADM.1	Administrator guidance
	AGD_USR.1	User guidance
Life Cycle Support	ALC_DVS.1	Identification of security measures
	ALC_FLR.2	**Flaw Reporting Procedures**
	ALC_LCD.1	Developer defined life-cycle model
	ALC_TAT.1	Well-defined development tools
Tests	ATE_COV.2	Analysis of coverage
	ATE_DPT.2	**Testing: low-level design**
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing - sample
Vulnerability Assessment	**AVA_CCA_(EXP).2**	**Systematic cryptographic module covert channel analysis**
	AVA_MSU.2	Validation of analysis
	AVA_SOF.1	Strength of TOE security function evaluation
	AVA_VLA.3	**Moderately resistance**

III. General Information Instructions

(Back to TOC)

Instruction 5: Content and outline of a PP

(Back to TOC)

Title page

The Title Page will include the title, version and date of the PP. See Instruction 6 and Appendix E for details about the title page

1. Introduction to the PP

- 1.1 PP Identification
- 1.2 PP Overview of the PP
  - 1.2.1 General Environmental Characterization
- 1.3 Conventions – See instruction 13
- 1.4 Glossary of terms – See instruction 14
- 1.5 Document Organization

2. TOE Description – See instruction 2

- 2.1 Product type
- 2.2 Toe Definition
- 2.3 General TOE functionality
- 2.4 TOE Operational environment

3. Security Environment

- 3.1 Threats – See instruction 8
- 3.2 Organizational Security Policies – See instruction 9
- 3.3 Assumptions – See instruction 7

4. Security Objectives

- 4.1 TOE Security Objectives – See instruction 9
- 4.2 Environment Security Objectives - See instruction 10

5. IT Security Requirements

- 5.1 TOE Security Functional Requirements – See instructions 15-33
- 5.2 Security Requirements for the IT Environment - See instruction 10
- 5.3 TOE Security Assurance Requirements – See instruction 4

6. Rationale

- 6.1 Rationale foe TOE Security Objectives - See Appendix A
- 6.2 Rationale for the security objectives and security functional requirements for the environment
- 6.3 Rationale for TOE Security Requirements - See Appendix B
- 6.4 Rationale for assurance requirements
- 6.5 Rational for strength of function claim
- 6.6 Rational for satisfying all dependencies
- 6.7 Rational for explicit requirements
- 5.8 Rationale for not addressing consistency instructions

7. Appendices:

A. References
B. Glossary - See **instruction 14**
C. Acronyms
D. Robustness Environment Characterization – See instruction 1

Instruction 6: Format for the title page of a PP
(Back to TOC)

In general, whole numbers (starting with 1) will be reserved for NIAP validated profiles, and decimal numbers (starting with 0.1) will be used for draft profiles, which are released for review outside of the immediate development team. The team may use finer granularity for its internal coordination and tracking purposes

NIAP Validated profile will be whole numbers starting with 1 and increased by 1 for each new revision that get NIAP validated. Examples will be “Version 1”, “Version 3”, etc not “Version 1.0” or “Version 3.0.”

Draft profiles will start decimal numbers starting with 0.1 and increased by .1 for each new draft released outside of the development team. Examples will be “Version 0.1”, or “Version 0.3”. Drafts are documents that have been written and are under going various stages of review. Once a draft is written and released for the first review, it will be labeled “Version 0.1”. If no changes are required during a review the version number will remain the same, however if it is determined that changes are required the draft version number will be increase by .1 indicating the changes were made and the review process continues (even if it is back to the same review step).

When it is required to update a NIAP validated PP, the updated drafts will be numbered “Version 1.1”, or “Version 1.2”, etc. Once the NIAP validates the new draft, it will get a new NIAP validated whole number 2, 3, etc.

In addition to the version number, the profile will contain a title of the profile and the date of the proposed version. The format of the date will be yyyymmdd. The title of the document should be provided in the following format "U.S. Government PP for (technology) used in (Robustness Level) Environments." Since we are now in a joint NSA/NIST process all profile will be U.S. Government and not DoD specific.

See appendix E for the template that shall be used by the Profile Author. The author shall fill in the technology area, date, version number and use cover sheet for their Profile.

Instruction 7: Assumptions
(Back to TOC)

Assumptions (included in Section 3 of the PP) are defined as ***non-IT*** items that the TOE itself cannot implement or enforce. Assumptions should not be used to specify functional requirements on the IT environment; that should be done with a threat or policy statement. For instance, a valid assumption might be "All administrators will be trained in the secure administration of the TOE." The TOE has no control over whether the administrators are trained or not, so this is a valid assumption. An invalid assumption might be "All users are authenticated before taking any action on the TOE." Since the TOE (or IT environment) could implement this, it is not a valid assumption.

In addition, it is useful to readers of the PP to list assumptions necessary for the TOE to work correctly.
From the initial review of several PP, the PPRB identified a few assumptions that seem to be frequently specified by PP authors. The text below proposes consistent names and descriptions for these commonly included assumptions. Note that not all assumptions will be valid for all PPs. PP authors need to determine if whether specific assumptions apply to the TOE being described in the PP.

	Text
A.NO_GENERAL_PURPOSE[4]	The administrator ensures there are no general-purpose computing or storage repository capabilities (e.g., compilers, editors, or user applications) available on the TOE.
A.PHYSICAL	It is assumed that the IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE.

[4] This assumption should be used only on "server"-type TOEs that should have no general-purpose functionality available to untrusted users. It makes sense, for example, for a firewall or a router, but does not make sense for an operating system or someone's desktop computer.

Instruction 8: Describing Threats
(Back to TOC)

Threat statements define threats – either to the assets that the TOE protects or to the TOE itself – That the TOE addresses in whole or in part (e.g., "Users might attempt access to resources to which they have no permission"). Threats (included in section 3 of the PP) are stated as risks to security that the TOE will mitigate or eliminate. Therefore, threat statements must **not** include situations in which the TOE plays no part (i.e., those that are completely addressed by the environment), threats the TOE cannot recognize (e.g., the TOE may be incorrectly configured), or threats to the TOE itself that would not exist without the TOE (e.g., the TOE may contain Trojan horses).

The PPRB recognized the importance of a clear understanding of the basis for specifying appropriate threats for a given robustness level and therefore, requires the inclusion in all PPs, a discussion that will establish the context of how to formulate applicable threats for a given robustness level. The following text should be included in section 3 of all PPs to explain to PP authors and reviewers, how the itemized threats as described in the TSE section were formulated.

Text for Describing the Threat Environment

Threat Agent Characterization

In addition to helping define the robustness appropriate for a given environment, the threat agent is a key component of the formal threat statements in the PP. Threat agents are typically characterized by a number of factors such as *expertise*, *available resources*, and *motivation*. Because each robustness level is associated with a variety of environments, there are corresponding varieties of specific threat agents (that is, the threat agents will have different combinations of motivation, expertise, and available resources) that are valid for a given level of robustness. The following discussion explores the impact of each of the threat agent factors on the ability of the TOE to protect itself (that is, the robustness required of the TOE).

The *motivation* of the threat agent seems to be the primary factor of the three characteristics of threat agents outlined above. Given the same expertise and set of resources, an attacker with low motivation may not be as likely to attempt to compromise the TOE. For example, an entity with no authorization to low value data none-the-less has low motivation to compromise the data; thus a basic robustness TOE should offer sufficient protection. Likewise, the fully authorized user with access to highly valued data similarly has low motivation to attempt to compromise the data, thus again a basic robustness TOE should be sufficient.

Unlike the motivation factor, however, the same can't be said for *expertise*. A threat agent with low motivation and low expertise is just as unlikely to attempt to compromise a TOE as an attacker with low motivation and high expertise; this is because the attacker with high expertise does not have the motivation to compromise the TOE even though

they may have the expertise to do so. The same argument can be made for *resources* as well.

Therefore, when assessing the robustness needed for a TOE, the motivation of threat agents should be considered a "high water mark". *That is, the robustness of the TOE should increase as the motivation of the threat agents increases.*

Having said that, the relationship between expertise and resources is somewhat more complicated. In general, if resources include factors other than just raw processing power (money, for example), then expertise should be considered to be at the same "level" (low, medium, high, for example) as the resources because money can be used to purchase expertise. Expertise in some ways is different, because expertise in and of itself does not automatically procure resources. However, it may be plausible that someone with high expertise can procure the requisite amount of resources by virtue of that expertise (for example, hacking into a bank to obtain money in order to obtain other resources).
It may not make sense to distinguish between these two factors; in general, it appears that the only effect these may have is to lower the robustness requirements. For instance, suppose an organization determines that, because of the value of the resources processed by the TOE and the trustworthiness of the entities that can access the TOE, the motivation of those entities would be "medium". This normally indicates that a medium robustness TOE would be required because the likelihood that those entities would attempt to compromise the TOE to get at those resources is in the "medium" range. However, now suppose the organization determines that the entities (threat agents) that are the least trustworthy have no resources and are unsophisticated. In this case, even though those threat agents have medium motivation, the likelihood that they would be able to mount a successful attack on the TOE would be low, and so a basic robustness TOE may be sufficient to counter that threat.

It should be clear from this discussion that there is no "cookbook" or mathematical answer to the question of how to specify exactly the level of motivation, the amount of resources, and the degree of expertise for a threat agent so that the robustness level of TOEs facing those threat agents can be rigorously determined. However, an organization can look at combinations of these factors and obtain a good understanding of the likelihood of a successful attack being attempted against the TOE. Each organization wishing to procure a TOE must look at the threat factors applicable to their environment; discuss the issues raised in the previous paragraph; consult with appropriate accreditation authorities for input; and document their decision regarding likely threat agents in their environment.

The important general points we can make are:

- The motivation for the threat agent defines the upper bound with respect to the level of robustness required for the TOE
- A threat agent's expertise and/or resources that is "lower" than the threat agent's motivation (e.g., a threat agent with high motivation but little expertise and few resources) may lessen the robustness requirements for the TOE (see next point, however).

- The availability of attacks associated with high expertise and/or high availability of resources (for example, via the Internet or “hacker chat rooms”) introduces a problem when trying to define the expertise of, or resources available to, a threat agent.

Instruction 9: Threats, Policies, Objectives and Requirements
(Back to TOC)

Medium robustness PPs should contain relevant **threats, policies and associated objectives and requirements** for the medium robustness level, and use a consistent naming convention and description. The PPRB has formulated a list of **threats, policies, and objectives** that must be considered for all medium robustness TOEs, and a methodology for instantiating these in a PP. Each threat or policy has one or more objectives that address the stated threat or policy, and each objective in turn has requirement components associated with it that address the stated objective and mitigate or implement the threat or policy.

Unfortunately, cutting-and-pasting of all of these items without careful consideration is not appropriate. Reasons include:

- a threat may not apply to a technology;
- a threat or policy may be applicable but may need to be tailored in a technology-specific way; or
- although the threat may be applicable for the technology, the way in which it is countered, or the resources to which it applies, may be different depending on the technology. This might necessitate a change in the objective and/or requirement components; or
- some technologies may have threats that are not provided in this guidance that need to be countered, or policies that need to be met. For these additional threats or policies, additional objectives may need to be formulated, and requirements added.

Additionally, for most threat/objective/requirement mappings the rationale (how a set of objectives satisfies a threat or policy, and how a set of requirement components meets an objective) will have to be written "from scratch" to reflect the unique aspects of the technology. Some rational is included in this document for reference and possible use in medium robustness PPs. Care should be taken to review it to ensure its validity before it is included.

PP Creation Methodology Overview

In order to enhance consistency in writing PPs, the PPRB has formulated a methodology that can be used by PP authors in creating a substantial portion of the PP. There are several things to note about this methodology:

- This methodology has been used to produce quality PPs that are consistent with the PPRB guidance given in Table 7, *Applicable Threats, Policies, Objectives and Requirements for Medium Robustness TOEs*. **This does not mean that other methodologies cannot be used.** If the PP authors have a different approach that will yield a PP that is consistent with the PPRB guidance, they are welcome to use it.
- While the PP writing team may not use the methodology described below, they should still use the threats, objectives, and requirements listed in Table 7 to ensure consistency with other medium robustness TOEs.
- The following methodology is for the creation of significant parts of the PP.

However, additional work will have to be done by the PP writing team to complete the document.

It is critical in writing a PP that the requirements support the objectives and either mitigate the threats, or implement the policies stated in the PP. The CC framework calls for this to be documented in "*rationale*" sections: one detailing how the objectives (and associated requirements) mitigate a threat or implement a policy, and one detailing how the requirements implement the objectives (see Instruction 12 for more information on writing the Rationale sections). It is important to note that because the threat/policy to objective rationale section has to detail *how the applicable requirements from the objective mitigate the threat (or implement the policy)*, it is important for the PP authors to "keep track" of how the threats/polices map to objectives, and what requirements from those objectives relate to the threat/policies.

The PPRB has found that using a spreadsheet to keep track of this information is helpful. Although such a spreadsheet *is not* part of the PP itself, it can be a useful tool for PP authors in tracking the association between threats/policies, objectives, and requirements. In Appendix C of this guidance a spreadsheet has been prepared that has been "pre-loaded" with the information in Table 7. The PP authors can update this spreadsheet as they are working through the steps in the methodology so that when they are ready to write the rationale sections, they can ensure that they have accurately captured the relationship between all three "levels" in the requirements decomposition (those three levels being: threats/policies, objectives, and requirements).

Using Table 7: Applicable Threats, Policies, Objectives and Requirements for Medium Robustness TOEs

Table 7 consists of three columns. The first column indicates the threats and policies that the PP author must include in their medium robustness PP. Since the PP author does not derive or generate policies, all policies presented in the PP must reference the appropriate official policy document (e.g., DODI 8500.2, Committee on National Security Systems Policy, PDD 63). Each of the threats is mitigated by one or more objectives; likewise, each of the policies is implement by one or more objectives. For each threat/policy, the objective or objectives that mitigate/implement it are listed in the second column. Note that the same objective may be listed more than once in this second column, depending on how many of the threats/policies it applies to.

Each objective is implemented by one or more requirements ("components" in CC terminology). While multiple requirement components may be used to implement an entire objective, in some cases only a subset of those requirement components are used to counter a specific threat or implement a specific policy. This is reflected in the table by listing in column 3 only those requirements that apply to the particular threat or policy in column 1.

For instance, from Table 7 the PPRB suggests that O.ROBUST_TOE_ACCESS be implemented by FIA_AFL.1, FIA_ATD.1, FIA_UID, FIA_UAU, FTA_SSL.1, FTA_SSL.2, FTA_SSL.3, and AVA_SOF. O.ROBUST_TOE_ACCESS partially

mitigates the T.MASQUERADE threat, fully mitigates the T.UNATTENDED_SESSION threat, and partially implements the P.ACCOUNTABILITY policy. However, not all of the requirements associated with O.ROBUST_TOE_ACCESS are applicable to all of the threats and policies that O.ROBUST_TOE_ACCESS is associated with (e.g., only the FIA_UID component of O.ROBUST_TOE_ACCESS is used to implement P.ACCOUNTABILITY). This is why there may be different sets of requirements listed in column 3 for the same objective.

The last column of Table 7 contains notes on the information in that row. It may draw attention to the threat/policy, the objective, or the requirement. Where the PPRB is recommending specific text (e.g., an assignment, selection or refinement) be used for a requirement, it may refer the PP authors to another Instruction that contains the text the PP authors should use.

The PPRB suggests that the PP authors make a "working copy" of Table 7 so that if threats/policies are added, objectives added or changed, or when requirements are added or tailored, a centralized record can be maintained by modifying the copy of Table 7 appropriately. This will make it easier to create the CC-mandated tables that will appear in the PP in later steps in the methodology below. It is important to note that the only difference between this working copy of Table 7 and the Excel spreadsheet mentioned above and contained in Appendix C is that the Excel spreadsheet does not have the text associated with the threats and objectives, so that it can be more easily be viewed "all at once".

PP Creation Methodology

The methodology for incorporating the information in Table 7 into a PP is described in the following steps. The overall approach is for the PP author to start at the beginning of Table 7 and address the first threat, then the objectives that apply to that threat, and finally the components from those objectives that mitigate the threat. The PP authors then address the next threat-objective-component "thread" until all threats and policies have been addressed.[5] After the PP authors ensure that the technology-specific details are covered, the PP material (various tables) is created and the rationale written. The details of this process is as follows:

1. The PP authors select the first (or next, for subsequent iterations) threat or policy provided in the Table 7. *Applicable Threats, Policies, Objectives and Requirements for Medium Robustness TOEs*. They should review the threat/policy statement to ensure its applicability to the subject PP. Most threats/policies will apply directly to the technology being specified in the PP; if there are technology-specific aspects to a threat the PP authors should capture these aspects in the threat-to-objective rationale (see step 11) rather than try to create a new threat. Although a threat/policy may have to be tailored for a specific technology, this should be rare. Most threats/policies in Table 7 are sufficient so that no tailoring is necessary.

[5] While it is certainly feasible to perform the activity by first doing all of the threats/policies, then doing all of the objectives, and then doing all of the requirement components, the methodology described above appears to reduce iteration on the part of the PP authors.

2. If the threat/policy is not applicable to the technology, a short justification will need to be included in Table 2, *Medium Robustness Threats **Not** Applicable to the TOE*. See Step 9 for placement of this table. It should be noted that placing a threat/policy from Table 7 into this category should be rare. The PP authors must be careful to distinguish threats that really don't apply because of the nature of the technology from threats that can't be countered because current instantiations of the technology do not include the required features.

3. If the threat/policy is applicable, then the objectives associated with the threat/policy in the table should be examined for validity. Note that the same objective may apply to multiple threats/policies, and thus may appear multiple times in the table (for example, O.RESIDUAL_INFORMATION is associated with T.AUDIT_COMPROMISE, T.RESIDUAL_DATA, and T.MALICIOUS_TSF_COMPROMISE). This means the PP authors will have to ensure that any text added or modified for an objective is applicable for all threats/policies to which that objective applies. In some cases, new objectives may need to be created; if so, the PP authors should ensure that the objective statements are consistent (with respect to format and level of detail) with those in the table.

4. Finally, the requirements components associated with each objective for the given threat/policy should be examined. The last column of Table 7 makes reference to some Instructions containing actual requirement component text (for example FAU_GEN.1-NIAP-407/0410 and Instruction 16 of this document); the PPRB feels that this text should be included in the PP verbatim unless there is good justification for not doing so. Such text includes assignments, selections, etc. that is important to keep intact from a consistency perspective across all medium robustness PPs. In reviewing a medium robustness PP the PPRB will note requirements that were not included verbatim, and will ask the PP authors for a rationale for omitting the recommended text. The PP authors should therefore ensure that when the decision is made to omit the recommend requirement text, a justification for this action is written and submitted with the PP for review by the PPRB.

   The PP authors should check to ensure that, for each requirement component chosen, the requirement component (1) applies to the objective and (2) mitigates some aspect of the threat/policy. The PP authors may want to make notes for the rationale section while they are doing this (see steps x and y, below). This step will be the most time consuming, and the PP authors may find they need to create new objectives, new threats/policies, etc. in the course of selecting components.

5. The PP authors then repeat steps 1 through 4 for each of the threats and policies listed in Table 7.

6. After the PP authors have gone through all of the threats and policies in the table, they need to consider if there are any technology-specific threats that need to be met by compliant TOEs. When considering such threats, the PP authors should consider whether the threat is appropriate for the medium

robustness environment and whether the threat may be covered by an existing threat or policy. If the PP authors identify technology-specific aspects of an existing medium robustness threat, the PP authors should ensure that those aspects are captured in the threat-to-objective rationale statement (see step 11) as opposed to creating *new* technology-specific threats. For each new threat that is created, the objectives that will counter that threat should be either picked from existing objectives or (more likely) created by the PP authors, and components picked that meet the objective and mitigate the threat. The policies identified in Table 7 should be sufficient for all medium robustness TOEs. It is generally not necessary to create additional technology-specific policies because the requirements that would be derived from such policies would already be covered by existing threats and policies.

7. After performing the above steps, the PP authors should review the components to ensure that all desired functionality is included. If it is determined that some desired functionality is omitted, the PP authors should review the threat and policy statements to determine if the functionality is needed to counter one of the existing threats or implement one of the existing policies. In the unlikely event that no applicable threat or policy is found, the PP authors should devise a threat or policy statement (and associated objective) to which the functionality would apply, and then choose the appropriate components from the CC to require the functionality.

   At the completion of step 7 all of the threats, policies, objectives, and requirements for the technology should be identified. If the PP authors have been modifying the working copy of Table 7 with updates to the threats, policies, objectives, and requirement component identifiers, the modified table will aid the team in their next tasks: creation of the threat, policy, and objective tables, and creation of the rationale.

8. The PP writing team should next construct a threat table (like Table 1 below) for the TOE environment section of the PP that details all of the threats that apply to the TOE. The table should consist of each threat label, followed by the threat text. The threats should be in alphabetical order. A sample format follows:

Table 1 Medium Robustness Applicable Threats

Threat Name	Threat Definition
T.AUDIT_COMPROMISE	A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.

A similarly formatted table should be created for the policies and included in

the TOE environment section.

The PP authors should also introduce the threat table with the following text:

The following threats are addressed by the TOE and should be read in conjunction with the threat rationale section. There are other threats that the TOE does not address (e.g., malicious developer inserting a backdoor into the TOE) and it is up to a site to determine how these types of threats apply to its environment.

9. For those threats found to be not applicable to the TOE because the threat does not "make sense" for the technology area (see step 2 above), the PP authors should construct a table such as Table 2 below that details the threat label, the text of the threat, and a short rationale detailing why the threat is not applicable for the technology. This table is not included in the PP, but provides a justification that the threats considered for that specific technology are commensurate with those in other medium robustness PPs.

Table 2 Medium Robustness Threats NOT Applicable to the TOE

Threat Name	Threat Definition	Rationale for NOT Including this Threat
T.ADMIN_ERROR	An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.	There are no administrators on compliant TOEs.

10. The PP writing team should then construct an table of objectives for the TOE Objectives section of the PP that details all of the objectives. The objectives should be drawn from two sources. First, for each assumption on the IT environment (see Instruction 7) an objective for the IT environment should be created (see Table 3). Additionally, if a threat is mitigated (or a policy implemented) by both the TOE and the IT environment, then an objective for the environment (in addition to the objective(s) for the TOE listed in Table 7) should be created for each of these. The environmental objectives should have a tag of "OE.*assumption_tag*", where *assumption_tag* is the tag associated with the assumption. For example, for the assumptions given in Instruction 7:

Table 3 Objectives for the IT Environment

IT Environment Objective Name	Environment Objective Definition

OE.NO_GENERAL_ PURPOSE	There will be no general-purpose computing or storage repository capabilities (e.g., compilers, editors, or user applications) available on the TOE.
OE.PHYSICAL	Physical security will be provided within the domain for the value of the IT assets protected by the operating system and the value of the stored, processed, and transmitted information.

Second, all objectives generated in steps 1 through 6 need to be captured in an objective table (in alphabetical order). The format is similar and is shown in Table 3:

Table 4 TOE Objectives

Objective Name	Objective Definition
O. ROBUST_ADMIN_GUIDANCE	The TOE will provide administrators with the necessary information for secure delivery and management.
O.AUDIT_GENERATION	The TOE will provide the capability to detect and create records of security-relevant events associated with users.

11. The threat/policy-objective rationale section should be created next. In writing this rationale, the PP authors should use the format shown in Table 5.

Table 5 Threat/Policy to Objective Rationale

Threat/Policy	Objectives Addressing the Threat	Rationale
T.ADMIN_ERROR An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.	O. ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	O. ADMIN_GUIDANCE (ADO_DEL.2, ADO_IGS.1, AGD_ADM.1, AGD_USR.1, AVA_MSU.2) help to mitigate this threat by ensuring the TOE administrators have guidance that instructs them how to administer the TOE in a secure manner and to provide the administrator with instructions to ensure the TOE was not corrupted during the delivery process. Having this guidance helps to reduce the mistakes that an administrator might make that could cause the TOE to be configured in a way that is insecure.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.AUDIT_COMPROMISE A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.	O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information. O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated. O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.	O.AUDIT_PROTECTION (FAU.SAR.2, FAU_STG.1-NIAP-0429, FAU_STG.3, FAU_STG.NIAP-0414-1, FMT_SMF.1) contributes to mitigating this threat by controlling access to the audit trail. The auditor and any trusted IT entities performing IDS-like functions are the only ones allowed to read the audit trail. No one is allowed to modify audit records, and the Auditor is the only one allowed to delete audit records in the audit trail. The TOE has the capability to prevent auditable actions from occurring if the audit trail is full, and of notifying an administrator if the audit trail is approaching its capacity. In addition, the TOE has the capability to restore audit data corrupted by the attacker. O.RESIDUAL_INFORMATION (FDP.RIP.2) prevents a user not authorized to read the audit trail from access to audit information that might otherwise be persistent in a TOE resource (e.g., memory). By ensuring the TOE prevents residual information in a resource, audit information will not become available to any user or process except those explicitly authorized for that data. O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the audit trail. Likewise, ensuring that the functions that protect the audit trail are always invoked is also critical to the mitigation of this threat.

The first two columns of this table are identical to the first two columns of Table 7. The rationale should address how each objective contributes to mitigating the threat or implementing the policy, and the applicable components from each objective should be identified. In Appendix A of this Manual we have supplied sample rational for several threats.

12. The PP authors should then write the objective/requirement component rationale. The format for this rationale should be as is shown in Table 6.

Table 6 Objective to Requirements Rationale

Objectives	Requirements addressing Objectives	Rational
O. ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	ADO_DEL.2 AGD_ADM.1 AVA_MSU.2 ADO_IGS.1 AGD_USR.1	ADO_DEL.2 ensures that the administrator is provided documentation that instructs them how to ensure the delivery of the TOE, in whole or in parts, has not been tampered with or corrupted during delivery. This requirement ensures the administrator has the ability to begin their TOE installation with a *clean* (e.g., malicious code has not been inserted once it has left the developer's control) version of the TOE, which is necessary for secure management of the TOE. The ADO_IGS.1 requirement ensures the administrator has the information necessary to install the TOE in the evaluated configuration. Often times a vendor's product contains software that is not part of the TOE and has not been evaluated. The Installation, Generation and Startup (IGS) documentation ensures that once the administrator has followed the installation and configuration guidance the result is a TOE in a secure configuration. The AGD_ADM.1 requirement mandates the developer provide the administrator with guidance on how to operate the TOE in a secure manner. This includes describing the interfaces the administrator uses in managing the TOE, security parameters that are configurable by the administrator, how to configure the TOE's ruleset and the implications of any dependencies of individual rules. The documentation also provides a description of how to setup and review the auditing features of the TOE. The AGD_USR.1 is intended for non-administrative users, but could be used to provide guidance on security that is common to both administrators and non-administrators (e.g., password management guidelines). Since the non-administrative users of this TOE are limited to relying parties it is expected that the user guidance would discuss how the data validation authentication mechanism is used, and any instructions on authenticating to the TOE. The description of the use of these mechanisms would not have to be repeated in the administrator's guide. AVA_MSU.2 ensures that the guidance documentation is complete and can be followed unambiguously to ensure the TOE is not misconfigured in an insecure state due to confusing guidance.

Objectives	Requirements addressing Objectives	Rational
O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security-relevant events associated with users.	FAU_GEN.1-NIAP-0407 FAU_GEN.2-NIAP-0410 FIA_USB.1 FAU_SEL.1-NIAP-0407	FAU_GEN.1-NIAP-0407 defines the set of events that the TOE must be capable of recording. This requirement ensures that an administrator has the ability to audit any security relevant event that takes place in the TOE. This requirement also defines the information that must be contained in the audit record for each auditable event. There is a minimum of information that must be present in every audit record and this requirement defines that, as well as the additional information that must be recorded for each auditable event. This requirement also places a requirement on the level of detail that is recorded on any additional security functional requirements an ST author adds to this PP. FAU_GEN.2-NIAP-410 ensures that the audit records associate a user identity with the auditable event. Although the FIA_ATD.1(*) requirements mandate that a "userid" be used to represent a user identity, the TOE developer is able to associate different types of userids with different users in order to meet this objective. FAU_SEL.1-NIAP-0407 allows the selected administrator(s) to configure which auditable events will be recorded in the audit trail. This provides the administrator with the flexibility in recording only those events that are deemed necessary by site policy, thus reducing the amount of resources consumed by the audit mechanism and providing the ability to focus on the actions of an individual user. In addition, the requirement has been refined to require that the audit event selection function is configurable during run-time to ensure the TOE is able to capture security-relevant events given changes in threat conditions. FIA_USB.1 plays a role is satisfying this objective by requiring a binding of security attributes associated with users that are authenticated with the subjects that represent them in the TOE. This only applies to authenticated users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the subject that causes an audit record to be generated (anonymous relying parties).

As with the previous rationale, the objective/component rationale should address how each component contributes to satisfying the objective. In Appendix B of this Manual we have supplied sample rational for several objectives.

In writing the rationale sections the PP authors may discover that a threat is not mitigated to the extent desired, or that an objective is not fully met. The PP authors will have to resolve these discrepancies by adjusting the threat/policy/objective statement or by adjusting component or element text, or by including a new component.

Table 7 Applicable Threats, Policies, Objectives and Requirements for Medium Robustness TOEs

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T. ADMIN_ ERROR An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.	O. ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	ADO_DEL.2, ADO_IGS.1, AGD_ADM.1, AGD_USR.1, AVA_MSU.2	
	O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	FMT_SMR	See T.ADMIN_ROGUE. We recommend that roles be included in medium robustness PPs for at least cryptography (if cryptography is included) and all other functions. If T.ADMIN_ROGUE does not "make sense" for a technology, then O.ADMIN_ROLE will not have to be achieved by TOEs for that technology area.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
	O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	FMT_MTD	Any FMT_MTD iterations that allow an administrator to review configuration settings for the security mechanisms are considered as contributing to the avoidance of errors.
T.ADMIN_ROGUE An administrator's intentions may become malicious resulting in user or TSF data being compromised.	O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	FMT_SMR	The component from the FMT_SMR family should be either FMT_SMR.2 or FMT_SMR.3, depending on technology and TAL guidance. If crypto is included, then there must be a crypto role. There also must be at least two administrative roles regardless of whether crypto is included or not. For instance, "auditor" and "administrator" roles; "operator" and "administrator" roles; "security administrator" and "administrator" roles. Having chosen the roles, the PP authors must ensure that all components in the PP that specify "administrator" or "authorized administrator" be changed to specifically call out the appropriate role or roles for that particular function. If different roles have different functionality for the same mechanism, the PP author may need to iterate the requirement for each applicable role. In the text required below, this has been identified by the notation <role administrator>; This should be replaced by the PP authors with the appropriate role.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.AUDIT_COMPROMISE A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.	O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information.	FMT_MOF, FAU_SAR.2, FAU_STG.1-NIAP-0429, FAU_STG.3, FAU_STG.NIAP-0414-1	There should exist an iteration of FMT_MOF that applies to the audit functionality of the system; that iteration should be associated with this threat/objective combination. For FAU_STG.1-NIAP-0429 the PP authors should include the text in Instruction 18. For FAU_STG.3, the PP authors should include the text written in Instruction 11. FAU_STG.NIAP-0414-1 provides functionality similar to FAU_STG.4 and should be used instead of FAU_STG.4; the PP authors should include the text written in Instruction 20.
	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	FDP_RIP.2	
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	FPT_SEP, FPT_RVM.1	If crypto is included in the TOE, then FPT_SEP.2 (see Instruction 23 below) should be used; otherwise, use FPT_SEP.1. As noted in Instruction 2, inclusion of FPT_SEP as a requirement on the TOE means that hardware the TOE relies on to implement its security functionality has to be part of the TOE.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.CRYPTO_COMPROMISE A malicious user or process may cause key, data or executable code associated with the cryptographic functionality to be inappropriately accessed (viewed, modified, or deleted), thus compromising the cryptographic mechanisms and the data protected by those mechanisms.	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.		See Instruction 23 for cryptography
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.	FPT_SEP, FPT_RVM.1	See Instruction 23 for FPT_SEP.2.
	O.DOCUMENT_KEY_LEAKAGE The bandwidth of channels that can be used to compromise key material shall be documented.	AVA_CCA_(EXP).2	See Instruction 23 .
T.EAVESDROP A malicious user or process may observe or modify user or TSF data transmitted between physically separated parts of the TOE.	O.PROTECT_IN_TRANSIT The TSF shall protect user and TSF data when it is in transit from one portion of a distributed TOE to another.	FDP_ITT.1, FPT_ITT.1	Both user and TSF data need to be protected in transit for modification and disclosure, so both components are needed. This protection should be done with cryptography; see Instruction 23. The PP author may want to specify a particular encryption operation to be used for the requirements by including the in the refinement a reference to an appropriate iteration of FCS_COP.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.MASQUERADE A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources.	O. ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	FIA_AFL.1, FIA_ATD.1, FIA_UID, FIA_UAU, FTA_TSE.1, AVA_SOF	This is an area that different technologies may address in different ways; some modification of the threat and objective may be necessary. The choice of the applicable FIA requirements will also depend on technology-specific concerns. If TOEs in a particular technology depend on other entities (e.g., a certificate authority, DNS server) in order to perform a security function, then the authors need to include FPT_ITC (used to satisfy the O.TRUSTED_PATH) in order to protect the communication between the TOE and the external entities; appropriate environmental requirements must be included in the PP as well. For FIA_AFL.1, see Instruction 26 for the required text. See Instruction 11, Interpretation I-0375, for information on specifying FAU_UAU requirements for a single authentication mechanism. See Instruction 35 for FTA_TSE.1.
T.FLAWED_DESIGN Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious user or program.	O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	ACM_AUT.1, ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1	

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
	O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.	ADV_FSP_(EXP).1, ADV_HLD_(EXP).1, ADV_INT_(EXP).1, ADV_LLD_(EXP).1, ADV_ARC_(EXP).1, ADV_RCR.1, ADV_SPM.1	If cryptography is included, see Instruction 23 regarding ADV_SPM.1.
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	AVA_VLA.3	
T.FLAWED_IMPLEMENTATION Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious user or program.	O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	ACM_AUT.1, ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1	
	O.SOUND_IMPLEMENTATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented.	ADV_IMP.2, ADV_LLD_(EXP).1, ADV_RCR.1, ADV_INT_(EXP).1, ADV_ARC_(EXP).1, ALC_TAT.1	
	O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	ATE_COV.2, ATE_FUN.1, ATE_DPT.2, ATE_IND.2	

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	AVA_VLA.3	
T.POOR_TEST Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered thereby causing potential security vulnerabilities.	O.CORRECT_ TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	FPT_TST	If cryptography is included then self-test for that functionality must be specified through iteration of FPT_TST; see Instruction 23. See Instruction 30 for guidance on FPT_TST for non-cryptographic portions of the TOE.
	O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	ATE_COV.2, ATE_FUN.1, ATE_DPT.2, ATE_IND.2	
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	AVA_VLA.3	

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.REPLAY A user may gain inappropriate access to the TOE by replaying authentication information, or may cause the TOE to be inappropriately configured by replaying TSF data or security attributes (e.g., captured as transmitted during the course of legitimate use).	O.REPLAY_DETECTION The TOE will provide a means to detect and reject the replay of authentication data as well as other TSF data and security attributes.	FPT_RPL.1	See Instruction 28 for specific wording for FPT_RPL.1.
T.RESIDUAL_DATA A user or process may gain unauthorized access to data through reallocation of TOE resources from one user or process to another.	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	FDP_RIP.2, FCS_CKM	The FCS requirements should be included to place additional requirements on keys if cryptography is included in the TOE; see Instruction 23.
T.RESOURCE_EXHAUSTION A malicious process or user may block others from system resources (e.g., *example of resources that apply to technology*) via a resource exhaustion denial of service attack.	O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust *<specific types of resources, which the TOE protects>* resources provided by the TOE (e.g., *examples of resources that apply to technology*).	FRU_RSA.1, FMT_MTD.2, FMT_MOF.1	See Instruction 32 for more information on how this should be instantiated in the PP. The PP author should replace the italicized text in the threat and objective with technology-specific information, and iterate FRU_RSA.1, FMT_MTD.2, and FMT_MOF.1 as required.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.SPOOFING A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	FTP_TRP, FTP_ITC	Since "users" include both human users and IT entities, the PP author should consider which of the two components (or both) might be necessary for the technology. For instance, if it is not required that IT entities authenticate to the TOE, then FTP_ITC may not need to be included. If the PP authors want to specify encryption as being the means to implement these requirements, see Instruction 23 for wording for these components.
T.MALICIOUS_TSF_ COMPROMISE A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	FDP_RIP.2, FCS_CKM	The FCS requirements should be included to place additional requirements on keys if cryptography is included in the TOE; see Instruction 23.
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	FPT_SEP, FPT_RVM.1	See Instruction 2 for FPT_SEP.2.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
	O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	FMT_MTD.1, FMT_MSA.1, FMT_MOF.1 FMT_SMF.1	For MTD and MOF, the PP authors should group the data and functions according to 1) who has access and 2) the actions that the users can perform. The requirements should be iterated for each unique set of actions that are specified. It should be noted that for FMT_MSA.1, the attributes are defined with respect to a user data access control policy (FDP_ACC, FDP_IFC) and should not- be used for general "security attribute" restrictions. The requirement FMT_SMF.1 was introduced as an international interpretation. This requirement specifies functionality that must be provided to administrators of the TOE. If the PP author includes this requirement care must be taken to use the other FMT requirements to specify how the functionality is restricted and to which role the functionality is provided.
	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	FTA_TAB.1	See Instruction 34 for FTA_TAB.1.
	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	FTP_TRP, FTP_ITC	

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.UNATTENDED_SESSION A user may gain unauthorized access to an unattended session.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	FTA_SSL.1, FTA_SSL.2, FTA_SSL.3, AVA_SOF.1	FTA_SSL.3 is needed only if remote activity (e.g., remote administration) is included as required functionality for this technology.
T.UNAUTHORIZED_ACCESS A user may gain access to user data for which they are not authorized according to the TOE security policy.	O.MEDIATE The TOE must protect user data in accordance with its security policy.	FDP_ACC* FDP_ACF*, FDP_IFF*	This threat is one of the most technology-specific, and will likely require substantial modification to focus on the access control policy implemented in the technology. This applies only to user data (TSF data are covered by other threats). Additional objectives may need to be created, and the wording for O.MEDIATE will likely need to be modified. It may not be necessary to include the FDP_ACF, FDP_ACC* or the FDP_IFF* families. Other components from FDP might also be included, again dependent on the technology. See Instructions 24 and 25 for interpreted FDP_ACF and FDP_IFF requirements, respectively, to use as a baseline for the technology-specific requirements.
	O.USER_GUIDANCE The TOE will provide users with the information necessary to correctly use the security mechanisms.	AGD_USR.1	The AGD_USR.1 is intended for non-administrative users, but could be used to provide guidance on security that is common to both administrators and non-administrators (e.g., password management guidelines). Since the non-administrative users of this TOE are limited to relying parties it is expected that the user guidance would discuss the various security mechanisms and how to use them correctly.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
T.UNIDENTIFIED_ACTIONS The administrator may fail to notice potential security violations, thus limiting the administrator's ability to identify and take action against a possible security breach.	O.AUDIT_REVIEW The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.	FAU_SAA.1-NIAP-0407, FAU_ARP.1, FAU_ARP_ACK_(EXP).1, FAU_SAR.1, FAU_SAR.3	See Instruction 21 for information on including FAU_SAA.1-NIAP-0407, FAU_ARP.1, and FAU_ARP_ACK_(EXP).1.1 in the PP. For FAU_SAR.3, the first selection should be "searches and sorting" to indicate that the capability to both search and to sort on the criteria is desired. The assignment in FAU_SAR.3 should include at least user identity, date, and time; technology-specific information should be included by the PP Authors in this list as well.
T.UNKNOWN_STATE When the TOE is initially started or restarted after a failure, the security state of the TOE may be unknown.	O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed.	FPT_RCV.2	See Instruction 29 for the required wording for the FPT_RCV.2 component.
	O.CORRECT_ TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	FPT_TST	If cryptography is included then self-test for that functionality must be specified through iteration of FPT_TST; see Instruction 14. See Instruction 30 for guidance on FPT_TST for non-cryptographic portions of the TOE.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
	O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.	ADV_SPM.1	ADV_SPM.1 requires the developer to provide an informal model of the security policies of the TOE. Modeling these policies helps understand and reduce the unintended side effects that occur during the TOE's operation that might adversely affect the TOE's ability to enforce its security policies.
	O. ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	ADO_IGS.1, AGD_ADM.1	ADO_IGS.1, AGD_ADM.1, help to mitigate this threat by ensuring the TOE administrators have guidance that instructs them how to administer the TOE in a secure manner and to provide the administrator with instructions to ensure the TOE has been installed, generated and started up in a secure manner as intended by the developer. Having this guidance helps to reduce the mistakes that an administrator might make that could cause the TOE to be configured in a way that is insecure.
P.ACCESS_BANNER The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE. Reference: DODI 8500.2 Enclosure 4, Attachment 4 ECWM-1 and ECAN-1	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	FTA_TAB.1	See Instruction 34 for FTA_TAB.1.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
P.ACCOUNTABILITY The authorized users of the TOE shall be held accountable for their actions within the TOE. Reference: DODI 8500.2 Paragraph 5.12, Enclosure 4, Attachment 1,2,3, and 4, ECAT-2, ECRG-1, ECTP-1, ECAR-3, ECLC, etc.	O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security-relevant events associated with users.	FAU_GEN.1-NIAP-0407, FAU_GEN.2-NIAP-0410, FIA_USB.1, FAU_SEL.1-NIAP-0407	FAU_GEN.1-NIAP-0407 and FAU_GEN.2-NIAP-0410 should be included as indicated in Instruction 16; the audit event types and additional audit information should be included in a table and be will specific to the requirements in the finalized PP. See Instruction 27 for the text for FIA_USB.1. See Instruction 17 for FAU_SEL.1-NIAP-0407.
	O.TIME_STAMPS The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.	FPT_STM.1, FMT_MTD.1	There should be a FMT_MTD.1 iteration that covers setting the time that applies to this objective.
	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	FIA_UID	
P.ADMIN_ACCESS Administrators shall be able to administer the TOE both locally and remotely through protected communications channels.	O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	FMT_SMR.2	See Instruction 33 for notes on requiring remote administration as part of P.ADMIN_ACCESS.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
Reference: DODI 8500.2 Enclosure 4, Attachment 1,2, and 3 ECPA-1	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	FTP_TRP, FTP_ITC	
P.CRYPTOGRAPHY The TOE shall use NIST FIPS validated cryptography as a baseline with additional NSA-approved methods for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys), and for cryptographic operations (i.e.; encryption, decryption, signature, hashing, key exchange, and random number generation services). Reference: DODI 8500.2 Enclosure 3, Paragraph E3.2.4.3.3	O.CRYPTOGRAPHY The TOE shall use NIST FIPS 140-2 validated cryptographic services.	FCS_BCM FCS_CKM. FCS_COP	See Instruction 23 for a general discussion of cryptography and associated requirements. Note that PP Authors should contact the appropriate NSA personnel to ensure that the requirements specified for FCS_CKM and FCS_COP components are compatible with what is being required for other medium robustness TOEs. Only NIST FIPS validated cryptography (methods and implementations) are acceptable for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys) and cryptographic services (i.e.; encryption, decryption, signature, hashing, key exchange, and random number generation services). Key management systems must be NSA-approved. (DoDI 8500.2 section E2.4.3).
	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	FCS_CKM	See Instruction 23 for the text for these requirements.

Threat/Policy	Objectives Addressing the Threat	Requirements associated with Objectives addressing the Threat	Notes
P.VULNERABILITY_ ANALYSIS_TEST The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential. Reference: DODI 8500.2 Enclosure 4, Attachment 5 ECMT-1	O.VULNERABILITY_ANALYSIS_ TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	AVA_VLA.3	O.VULNERABILITY_ANALYSIS (AVA_VLA.3) satisfies this policy by ensuring that an independent analysis is performed on the TOE and penetration testing based on that analysis is performed. Having an independent party perform the analysis helps ensure objectivity and eliminates preconceived notions of the TOE's design and implementation that may otherwise affect the thoroughness of the analysis. The level of analysis and testing requires that an attacker with a moderate attack potential cannot compromise the TOE's ability to enforce its security policies.

Instruction 10: Specifying Requirements on the IT Environment
(Back to TOC)

The requirements on the IT environment for medium robustness TOEs are limited in scope. There may be cases where entire services (e.g., a Certificate Authority, time server) may be located external to the TOE that provides information to the TOE that the TOE uses to enforce its policy. This might require the use of requirements on the IT environment. The PPRB recommends that such requirements be specified when appropriate, and offers the following guidance to PP authors in determining when they should specify requirements on the IT environment, and how they should specify those requirements.

PP authors should be cognizant that requirements on the IT Environment (e.g., a Certificate Authority, time server) are not verified or validated by a NIAP lab as part of this TOE evaluation. The Lab will only verify and validate those SFRs that are identified as requirements for the TOE (i.e. those requirements included in SFR, Section 5.1 of the PP). The SFRs that are levied upon the IT environment are assumed to function correctly; these functions are not verified as part of the TOE evaluation. Therefore the Certifying and Accrediting officials may want to use NIAP certified products to address the requirements allocated to the IT environment.

For medium robustness STs/products to claim compliance with a medium robustness PP, all IT requirements (those of the TOE and those of the environment) will be verified and validated. STs must identify how those SFRs are being satisfied (i.e. those requirements included in SFRs in IT Security Requirements, Section 5 of the PP). In cases where the requirements for the environment are specified as SFRs (rather than only Security Objectives), those SFRs must be verified, either in conjunction with the TOE evaluation, or as part of some other evaluation. If the environment's SFRs are included in an evaluated product (at least at medium robustness) the evaluator may accept the correct functionality of that SFR (based on the products own evaluation) or they may elect to test the functionality. If the environment's SFRs are not included in a validated product, those SFR will be required to under go testing necessary to be verified and validated at medium robustness. The lab will also verify and validate all interfaces to SFRs in the environment. The PP author must ensure that the Degree of Compliance as specified in Instruction 15 describes the way in which the environment SFRs are assessed.

In general, if a TOE depends upon another IT entity in order for the TOE to enforce its security policies, then IT environmental requirements are used to specify the behavior expected from that IT entity. Some PPs have attempted to use Assumptions (as in Instruction 7) to deal with the dependencies a TOE has on other IT products; this is an incorrect use of assumptions. Other PPs have included threats or OSPs that are solely mitigated/implemented by objectives on the IT environment (which pull in the requirements on the IT environment); using threats/OSPs in this manner might be limitless, and obscures the functionality that is the subject of the PP. Therefore, such threats/OSPs (and their associated objectives and requirements for the IT environment) are not allowed for PPs written using this manual. In other words, any threat/OSP

included in a PP written using this manual must trace to at least one requirement on the TOE.

Specifying IT environment requirements affords the PP author the opportunity to state what security functionality is required of other IT products using the same requirements language as that used to specify the TOE's security functionality. Using the same language is important because it allows the end user to more easily ascertain whether IT products can work together to enforce a security policy.

When determining what requirements should be levied on the IT environment, the PP author considers what interaction the TOE will have with other IT entities and how that interaction may impact the TOE's ability to enforce its policies. If the TOE stores or obtains TSF data or security attributes from another IT entity, then the TOE has some security relevant dependency on that IT entity. If the TOE has a trust relationship with another IT entity, then the TOE probably has some dependency on that IT entity. The PP author considers the extent of the TOE's dependencies on that IT entity and determines what security functionality must be present in that IT entity to make it trustworthy from the perspective of the TOE.

One approach to determining the IT environment requirements would be to consider the IT entity as though it were part of the TOE. The PP author could then determine if the requirements levied on the TOE would apply to this "piece". The PP author then considers whether any additional requirements need to be specified on IT environment due to the nature of how the TOE depends on the trusted IT entity. For example, suppose that the TOE requires communication channels (FTP_ITC) with other external entities to be encrypted. The IT environment requirements should levy the same requirements as are on the TOE, including the encryption that is required (i.e., the FCS family).

> With respect to presentation, when writing IT environment requirements the PP author should replace the text TSF with the text IT environment. This makes sense because the TSF is not ensuring the functionality; rather it is the IT environment that is expected to ensure the specified behavior. Other adjustments (e.g., replacing "TSC" with "IT environment's Scope of Control") may have to be made to the components as well.

Instruction 11: Scheme Interpretations
(Back to TOC)

This CIM requires that where applicable (e.g., for "new" PPs) NIAP Interpretations (NIs) and International Interpretations are used in developing PPs. Practical application of the CC and CEM against different types of security products and systems, as well as within different security environments, results in the need for interpretations of the CC and the CEM, in order to clarify their meaning.

As increasing numbers of people use the CC, inconsistencies or ambiguities are found in the wording. In order to address these concerns, the CC Interpretations Management Board (CCIMB) was formed. Regular meetings of the CCIMB, comprising representatives from the member nations, result in formal Interpretations, which specify textual updates to the CC and CEM.

National schemes likewise make pronouncements on any inconsistencies or ambiguities found in the CC, and may issue their own interpretations to be used within their own scheme; within CCEVS, the NIAP Interpretations Board (NIB) creates interpretations. NIAP, like all schemes, forwards its final interpretations to the CCIMB for international concurrence in order to minimize the divergence among the schemes. However, because the list of interpretations, both NIAP and international, is ever increasing, it is impractical to attempt listing all final interpretations in this document; doing so would require constantly updating this document.

Within this document are some specific CC changes that the authors believe needed to be incorporated into PPs; these are presented as explicit requirements or refinements. Many of these suggested wording changes result from NIs, although many of these changes had not yet become international interpretations when this document was written. In such cases, within this document the PP author is reminded to check for an international interpretation that specifies the wording to be used, so that the new wording would not be considered an explicit requirement in need of justification.

If there is no international interpretation, then the PP author should check the NIs to see if there is specific wording supplied to be used within the PP; the rationale is simply that the new wording is the result of the NIAP interpretation.

Final International Interpretations can be found at:
http://www.commoncriteriaportal.org/public/expert/index.php?menu=5

Final NIAP Interpretations are available with other public NIB database entries at:
http://niap.nist.gov/cc-scheme/PUBLIC/index.html

Instruction 12: Rationale Section
(Back to TOC)

In this instruction the PPRB recommended that the PP authors spend a good deal of their effort in formulating detailed and comprehensive rationale. Writing rationale is sometimes difficult, but experience has shown that it is an important tool in producing high-quality PPs and offers the following points that PP authors should keep in mind while writing rationale.

The CC requires that a PP include rationale that demonstrates that the requirements satisfy the security objectives, and that those objectives counter the threats and implement the policies. The rationale serves two purposes. One purpose is to help the reader understand the intent of the requirements and objectives. The second purpose is that the process of writing a detailed rationale helps the PP author ensure that they have incorporated the appropriate requirements into the PP, and have made the proper selections and assignments within the requirements.

Since requirement language is written in English and typically consists of short concise statements, there is often room for interpretation. The PP author's intent may not be readily apparent in the requirements and they may be interpreted in a way that was not intended by the author. Having well-written rationale affords the PP author the opportunity to discuss what each requirement is attempting to achieve. The ultimate goal in writing a rationale is to communicate to the reader how the chosen requirements are intended to mitigate the associated threats, and implement the associated policies, and to what degree. Unfortunately, in an attempt to provide a different "view" of the system the CC includes the notion of security objectives, which provide a layer of indirection in achieving the ultimate goal of countering threats/implementing policies through requirements.

Requirements to Objective Rationale

One concern with the notion of security objectives is that currently a ST can claim conformance to a PP by demonstrating that the security objectives are satisfied. This means they do not necessarily have to include the same requirements. Since the objectives are also written in English and are usually written at a high general level, it leaves the security objectives open to interpretation and the result can be a PP conformant ST that does not meet the PP author's intent. By providing enough detail in the requirements to security objective rationale, the PP author can present the rationale in enough detail to ensure the intent of the objective is understood, making it more difficult for an ST author to claim conformance without satisfying the intent of the PP author. When writing the rationale that the requirements satisfy the objectives, the PP author should keep in mind the threats that are being addressed by the given objective and write the rationale for the requirements to security objectives so the reader can determine, in conjunction with the security objective to threat rationale to what degree the threats are being countered.

Objectives to Threat/Policy Rationale

When writing the security objectives to threat/policy rationale the PP author informs the reader to what extent a threat is being countered. The PP author should rely on the arguments made in the requirements to security objective rationale as the basis for making the argument that the threat is mitigated. It is acceptable, in fact desirable, to identify aspects of a threat that are not fully countered by the TOE. The threats provided in the PP guidance documents are somewhat generic and are written at a high level. The security objective to threat rationale should provide the details of what the TOE is protecting against. If there are technology specific aspects of the high level threats, then those specifics should be addressed in the rationale.

For example, consider the T.MASQUERADE threat from Table 7: "A user or process may masquerade as another entity in order to gain unauthorized access to data or TOE resources." The authors of the Biometrics PP wanted to address several specific biometric-related threats in the PP, such as:

- an imposter may use an artificial hand/fingerprint or other synthetic means to gain unauthorized access;

- an imposter may know that their biometric characteristics are very similar to an enrollee and attempt to masquerade as that individual.

Rather than creating several new threats, our recommended approach is to include T.MASQUERADE and O.TOE_ACCESS, and address these specific aspects of T.MASQUERADE in the rationale section for T.MASQUERADE to O.TOE_ACCESS.

Writing the security objective to threat rationale section is further complicated by the fact that typically more than one objective is used to mitigate a threat. In addition, different aspects of an objective may be used to mitigate different threats. This is because different requirements that are used to satisfy an objective are used to counter different threats. For example, the objective O.RESIDUAL_INFORMATION is satisfied by two requirements in the medium robustness Firewall PP: FDP.RIP.2 and FCS_CKM.4. The threat T.CRYPTO_COMPROMISE is partially mitigated by the objective O.RESIDUAL_INFORMATION, however, only the functionality provided by FCS_CKM.4 is discussed in the objective to threat rationale, since requirement ensures that cryptographic critical data will not be compromised by residing in resources that are not "cleaned" before being released to untrusted users. On the other hand, the threat T.AUDIT_COMPROMISE is partially mitigated by the objective O.RESIDUAL_INFORMATION, and only the functionality provided by FDP.RIP.2 is discussed in the rationale, FCS_CKM.4 does not contribute to satisfying the threat of a compromise of audit data occurring. To clarify exactly what is being addressed, the PPRB recommends that the requirement components applicable to a specific threat/policy be identified and associated with the objective; see the example of T.AUDIT_COMPROMISE in Table 5, Threats/Policy to Objective Rationale.

One of the reasons given above for writing good rationale is to help the PP author ensure they have included the appropriate CC components, and have made the appropriate assignments and selections within an element. When writing a PP, the author has a general idea of what family of requirements they want, but there may some indecision over the component that is chosen or what assignments and selections to make. Going through the exercise of making an argument of how and to what extent a threat is countered by a requirement or set of requirements forces the PP author to ensure they have the right requirements for what they are intending to protect against.

As an example, an early version of the firewall PP required functionality that locked a user's proxy session after a period of inactivity. The PP included FTA_SSL.1 and FTA_SSL.2 to mitigate the threat T.UNATTENDED_SESSION. These two components ensure that the user can initiate the locking of their session, and that after a time interval of inactivity the session is locked. After considering the threat and thinking how proxy sessions are used in a firewall, it was determined that these two components did not address remote sessions in a way that made sense. Therefore, FTA_SSL.3 was added, which requires that the remote session be terminated after a period of inactivity.

> Assignments may not be filled in correctly, or there may be assignments that need to be made that are not readily apparent. Writing good rationale can aid in identifying these areas as well. For example, the assignment of *time interval of inactivity* was modified in the FTA_SSL component. Originally this was left as an open assignment to be filled in by the ST author, which could have been any value the ST author deemed to be acceptable. After discussions about what was to be achieved with this requirement the assignment was changed to *administrator specified time period of inactivity*.

Instruction 13: Conventions
(Back to TOC)

Except for replacing United Kingdom spelling with American spelling, the notation, formatting, and conventions used in this PP are consistent with version 2.2 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP reader.

The notation, formatting, and conventions used in this PP are largely consistent with those used in version 2.2 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP user. The CC allows several operations to be performed on functional requirements; *refinement*, *selection*, *assignment*, and *iteration* are defined in paragraph 2.1.4 of Part 2 of the CC. Each of these operations is used in this PP.

The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by the word "Refinement" in **bold text** after the element number and the additional text in the requirement in bold text.

Example of Refinement:
Original:

> FMT_SMR.1.2 The TSF shall be able to associate users with roles.

Refinement:

> FMT_SMR.1.2 **Refinement:** The TSF shall be able to associate users with **defined security** roles.

The **selection** operation is used to select one or more options provided by the CC in stating a requirement. Selections that have been made by the PP authors are denoted by *italicized text* in brackets, selections to be filled in by the ST author appear in square brackets with an indication that a selection is to be made, [selection:]. The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password. Assignments that have been made by the PP authors are denoted by showing the value in square brackets, [Assignment_value], assignments to be filled in by the ST author appear in square brackets with an indication that an assignment is to be made [assignment:].

Example of Selection and Assignment operation:

Original:

> FMT_MTD.1.1 The TSF shall restrict the ability to [selection: *change_default, query, modify, delete, clear*, [assignment: *other operations*]] the [assignment: *list of TSF data*] to [assignment: *the authorised identified roles*].

Selection and Assignments made:

> FMT_MTD.1.1 The TSF shall restrict the ability to [*change_default, query, modify, delete, clear* [view]] the [security related data] to [authorized users].

The **iteration** operation is used when a component is repeated with varying operations. Iteration is denoted by showing the iteration number in parenthesis following the component identifier, (iteration_number).

Example of Iteration:
FAU_SAA.1(1) Potential violation analysis (non real-time)
Hierarchical to: No other components.
Dependencies: FAU_GEN.1 Audit data generation
FAU_SAA.1(1).1 The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.

As this PP was sponsored, in part by NSA, National Information Assurance Partnership (NIAP) interpretations are used and are presented with the NIAP interpretation number as part of the requirement identifier (e.g., **FAU_GEN.1-NIAP-0407** for Audit data generation).

The CC paradigm also allows protection profile and security target authors to create their own requirements. Such requirements are termed 'explicit requirements' and are permitted if the CC does not offer suitable requirements to meet the authors' needs. **Explicit requirements** must be identified and are required to use the CC class/family/component model in articulating the requirements. In this PP, explicit requirements will be indicated with the "(EXP)" following the component name. Application Notes are provided to help the developer, either to clarify the intent of a requirement, identify implementation choices, or to define "pass-fail" criteria for a requirement. For those components where Application Notes are appropriate, the Application Notes will follow the requirement component.

Example of Explicit Requirement:

> FDP_SDC_(EXP).1 Stored data change notification
> Dependencies: No dependencies
> FDP_SDC_(EXP).1.1 The TSF shall record the time and date of last change in data content.

NAMING CONVENTIONS

Assumptions: TOE security environment assumptions are given names beginning with "A." followed by a descriptive label all in caps -- e.g., A.ADMINISTRATION.

Threats: TOE security environment threats are given names beginning with "T." followed by a descriptive label all in caps-- e.g., T.SIGNAL_DETECT.

Policy Statements: Policy statements are given names beginning with "P." followed by a descriptive label all in caps-- e.g., P.PHYSICAL_ACCESS.

Security Objectives for the TOE: Security Objectives are given names beginning with "O." followed by a descriptive label all in caps-- e.g., O.ACCESS.

Security Objectives for both the IT Environment and Non-IT Environment: Security Objectives are given names beginning with "OE." followed by a descriptive label all in caps-- e.g., OE.ACCESS

Instruction 14: Glossary
(Back to TOC)

The glossary is used to define very basic concepts such as roles and responsibilities that are specified in PPs should be used consistently in all PPs. The independent definition and usage of redundant terms by multiple PP development teams leads to confusion amongst our target audiences of customers, vendors and evaluators.

The PPRB developed a set of term and definitions to be considered for inclusion in all PPs. The following list consists of terms that should be considered first by PP authors when trying to decide how best to describe their particular TOE and TOE environment. PP authors are dissuaded from developing new, redundant terminology and definitions when one of these terms may be adequate

Text

In the CC, many terms are defined in Section 2.3 of Part 1. The following are a subset of those definitions. They are listed here to aid the user of the PP being developed and should be included in the Glossary (Appendix B) of the PP.

> ***Access*** -- Interaction between an entity and an object that results in the flow or modification of data.
>
> ***Access Control*** -- Security service that controls the use of resources[6] and the disclosure and modification of data.[7]
>
> ***Accountability*** -- Property that allows activities in an IT system to be traced to the entity responsible for the activity.
>
> ***Administrator*** -- A user who has been specifically granted the authority to manage some portion or all of the TOE and whose actions may affect the TSP. Administrators may possess special privileges that provide capabilities to override portions of the TSP.
>
> ***Assurance*** -- A measure of confidence that the security features of an IT system are sufficient to enforce its' security policy.
>
> ***Asymmetric Cryptographic System*** -- A system involving two related transformations; one determined by a public key (the public transformation), and another determined by a private key (the private transformation) with the property that it is computationally infeasible to determine the private transformation (or the private key) from knowledge of the public transformation (and the public key).
>
> ***Asymmetric Key*** -- The corresponding public/private key pair needed to

[6] Hardware and software.

[7] Stored or communicated.

determine the behavior of the public/private transformations that comprise an asymmetric cryptographic system.

Attack -- An intentional act attempting to violate the security policy of an IT system.

Authentication -- Security measure that verifies a claimed identity.

Authentication data -- Information used to verify a claimed identity.

Authorization -- Permission, granted by an entity authorized to do so, to perform functions and access data.

Authorized user -- An authenticated user who may, in accordance with the TSP, perform an operation.

Availability -- Timely[8], reliable access to IT resources.

Compromise -- Violation of a security policy.

Confidentiality -- A security policy pertaining to disclosure of data.

Critical Security Parameters (CSP) -- Security-related information (e.g., cryptographic keys, authentication data such as passwords and pins, and cryptographic seeds) appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.

Cryptographic Administrator -- An authorized user who has been granted the authority to perform cryptographic initialization and management functions. These users are expected to use this authority only in the manner prescribed by the guidance given to them.

Cryptographic boundary -- An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.

Cryptographic key (key) -- A parameter used in conjunction with a cryptographic algorithm that determines [7]:

- the transformation of plaintext data into cipher text data,
- the transformation of cipher text data into plaintext data,
- a digital signature computed from data,
- the verification of a digital signature computed from data, or
- a digital authentication code computed from data.

Cryptographic Module -- The set of hardware, software, firmware, or some

[8] According to a defined metric.

combination thereof that implements cryptographic logic or processes, including cryptographic algorithms, and is contained within the cryptographic boundary of the module.

Cryptographic Module Security Policy -- A precise specification of the security rules under which a cryptographic module must operate, including the rules derived from the requirements of this PP and additional rules imposed by the vendor.

Defense-in-Depth (DID) -- A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.

Discretionary Access Control (DAC) -- A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. These controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

Embedded Cryptographic Module -- One that is built as an integral part of a larger and more general surrounding system (i.e., one that is not easily removable from the surrounding system).

Enclave -- A collection of entities under the control of a single authority and having a homogeneous security policy. They may be logical, or may be based on physical location and proximity.

Entity -- A subject, object, user or another IT device, which interacts with TOE objects, data, or resources.

External IT entity -- Any trusted Information Technology (IT) product or system, outside of the TOE, which may, in accordance with the TSP, perform an operation.

Identity -- A representation (e.g., a string) uniquely identifying an authorized user, which can either be the full or abbreviated name of that user or a pseudonym.

Integrity -- A security policy pertaining to the corruption of data and TSF mechanisms.

Integrity label -- A security attribute that represents the integrity level of a subject or an object. Integrity labels are used by the TOE as the basis for mandatory integrity control decisions.

Integrity level -- The combination of a hierarchical level and an optional set of non-hierarchical categories that represent the integrity of data.

Mandatory Access Control (MAC) -- A means of restricting access to objects based on subject and object sensitivity labels.[9]

[9] The Bell LaPadula model is an example of Mandatory Access Control

Mandatory Integrity Control (MIC) -- A means of restricting access to objects based on subject and object integrity labels.

Multilevel -- The ability to simultaneously handle (e.g., share, process) multiple levels of data, while allowing users at different sensitivity levels to access the system concurrently. The system permits each user to access only the data to which they are authorized access.

Named Object -- An object that exhibits all of the following characteristics:

- The object may be used to transfer information between subjects of differing user identities within the TSF.
- Subjects in the TOE must be able to request a specific instance of the object.
- The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to request the same instance of the object.

Non-Repudiation -- A security policy pertaining to providing one or more of the following:

- To the sender of data, proof of delivery to the intended recipient,
- To the recipient of data, proof of the identity of the user who sent the data.

Object -- An entity within the TSC that contains or receives information and upon which subjects perform operations.

Operating Environment -- The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls.

Operational key -- Key intended for protection of operational information or for the production or secure electrical transmissions of key streams.

Peer TOEs -- Mutually authenticated TOEs that interact to enforce a common security policy.

Public Object -- An object for which the TSF unconditionally permits all entities "read" access. Only the TSF or authorized administrators may create, delete, or modify the public objects.

Robustness -- A characterization of the strength of a security function, mechanism, service or solution, and the assurance (or confidence) that it is implemented and functioning correctly. DoD has three levels of robustness:

Basic: Security services and mechanisms that equate to good commercial practices.

Medium: Security services and mechanisms that provide for layering of additional safeguards above good commercial practices.

High: Security services and mechanisms that provide the most stringent

protection and rigorous security countermeasures.

Secure State -- Condition in which all TOE security policies are enforced.

Security attributes -- TSF data associated with subjects, objects, and users that are used for the enforcement of the TSP.

Security level -- The combination of a hierarchical classification and a set of non-hierarchical categories that represent the sensitivity of the information [10].

Sensitivity label -- A security attribute that represents the security level of an object and that describes the sensitivity (e.g. Classification) of the data in the object. Sensitivity labels are used by the TOE as the basis for mandatory access control decision.

Split key -- A variable that consists of two or more components that must be combined to form the operational key variable. The combining process excludes concatenation or interleaving of component variables.

Subject -- An entity within the TSC that causes operations to be performed.

Symmetric key -- A single, secret key used for both encryption and decryption in symmetric cryptographic algorithms.

Threat -- Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.

Threat Agent - Any human user or Information Technology (IT) product or system, which may attempt to violate the TSP and perform an unauthorized operation with the TOE.

User -- Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.

U*.S. Government PP* – A PP that follows the instructions in this CIM Manual.

Vulnerability -- A weakness that can be exploited to violate the TOE security policy.

Instruction 15: Degree of Compliance
(Back to TOC)

The PP is a statement of the author's requirements; it identifies the assumptions being made, the threats to be addressed, the objectives to be met, and the requirements to be enforced. Because PPs are written to be implementation-independent, some details are omitted, which may result in ambiguities concerning the intent of the author.

This situation can be ameliorated by the PP author specifying the degree of exactness with which STs must exhibit in order for the ST to legitimately claim compliance to the PP. The three degrees of exactness are Exact, Strict, or Demonstrable.

Exact conformance is expected to be used by those PP authors with the most stringent requirements that are to be expressed in a single manner. This approach to PP specification will limit the PPs/STs able to claim conformance to the PP purely on the basis of the wording used in the PP, rather than a technical ability to meet the security requirements. This would most likely be used in Request for Development in a product acquisition process.

Exact conformance is oriented to the PP-author who requires evidence that the requirements in the PP are met precisely and that any ST claiming conformance is an instantiation of the PP; there are to be no additions or modifications from the specification of the PP. Specifically:

- Either the security problem definition and objectives specified in the PP are to be duplicated in the ST, or the ST is to merely reference the appropriate sections in the PP.
- Alternative security requirement claims to those in the PP cannot be used in the ST.
- No additional (functional or assurance) security requirement claims can be made in the ST.
- All assignment and selection operations remaining in the PP are to be completed in the ST.

Strict conformance is expected to be used by those PP authors with vast experience of developing PPs, who again have requirements that must be adhered to in the manner specified. However, Strict Conformance permits the PP/ST author claiming compliance to the PP to add to those requirements, provided it is in a restrictive manner. i.e. the additional requirements cannot weaken the existing requirements, so hierarchical components can be used or additional components that build on those specified.

Strict conformance is oriented to the PP-author who requires evidence that the requirements in the PP are met precisely and that the ST is an instantiation of the PP. Specifically:

- The statements of the security problem definition and the objectives are to be consistent with those in the PP. These statements can be reworded using terminology with which the ST consumer will be conversant. However, the conformance rationale is to demonstrate that each aspect of the statements specified in the PP has been provided in the ST.
- The objectives for the operational environment can be modified providing the statement of security objectives in the ST is more restrictive that than that of the PP. This can include reassigning an objective specified for the environment in the PP to be a TOE objective in the ST.

- The SFRs specified in the ST must be a non-strict superset of the SFRs specified in the PP; i.e. the ST must claim the SFRs specified in the PP as a minimum, and no alternative requirements can be claimed in the place of a PP SFR.
- The SARs specified in the ST must be a non-strict superset of the SARs specified in the PP; i.e. the ST must claim SARs specified in the PP as a minimum, and no alternative requirements can be claimed in the place of a PP SAR.
- The additional requirement claims made in the ST must result in the specification of the TOE being more restrictive than that of the PP.
- The completion of operations must be consistent with that in the PP; either the same completion will be used in the ST as that in the PP or one that makes the requirement more restrictive (the rules of refinement apply).

If the PP author does not wish objectives for the environment to be reassigned as objectives of the TOE, he should:

a) consider whether it would be more appropriate to require "exact" conformance;
b) express the objective for the environment in such a way that it cannot be reworded as a TOE objective, whilst remaining consistent with that specified in the PP.
c) consider whether it would be permissible for the TOE to meet this objective provided it could be configured. i.e. the security function in the TOE meeting the requirement can be switched off through a configuration option without adversely affecting any other security functions of the TOE.

Demonstrable conformance allows a PP author to describe a common security problem to be solved and generic guidelines to the requirements necessary for its resolution, in the knowledge that there is likely to be multiple ways of specifying a resolution.

Demonstrable conformance is orientated to the PP-author who requires evidence that the ST/TOE is a suitable solution to the generic security problem described in the PP. Demonstrable conformance also caters for the ST author wishing to claim conformance to multiple PPs. Specifically:

- The SARs specified in the ST must be a non-strict superset of the SARs specified in the PP. i.e. the ST must claim SARs specified in the PP as a minimum, and no alternative requirements can be claimed in the place of a PP SAR.
- The ST, although ensuring all requirements specified in the PP are expressed in the ST, is able to use alternative SFRs taken from Part 2 where applicable. A rationale will be provided to explain how the set of requirements specified in the ST is consistent with that specified in the PP.
- The ST author may specify SFRs in addition to those required to meet the security problem defined in the PP, if they are necessary to meet the (extended) security problem defined in the ST.
- Any changes to the operational environment description will make the description more restrictive in the sense of refinement), or be as a result of moving an objective specified for the operational environment in the PP to become an objective for the TOE in the ST. A rationale will be provided to explain how the

operational environment described in the ST is consistent with that described in the PP.

- The completion of operations will be consistent with those in the PP; i.e the same completion is used in the ST as that in the PP or a completion that makes the requirement more restrictive (the rules of refinement apply). For example, if the PP author restricts the selection of four items in the component FAU_GEN.1.1b to two items in the PP. The ST can then only choose from the two in the PP, and not the other two. Nevertheless, the ST author may also add some audit events within the assignment in FAU_GEN.1.1c.

Specifying the degree of conformance

The PP author should specify the degree of exactness that STs must exhibit in order to claim compliance. It is recommended that the following statement be clearly included in Section 1, Introduction, of the PP along with the above definition of the degree of compliance required:

Any ST claiming compliance to this PP must do so in a/n [exact | strict | demonstrable] manner.

IV. Minimum CC Security Functional Requirement Instructions

A. Security Audit

(Back to TOC)

Instruction 16: Security Audit Generation

(Back to TOC)

The FAU_GEN.1-NIAP-0407 component should be structured in a consistent way. The events to be audited, as well as the information to be contained in the events, are currently presented in a variety of different ways. Further, the requirements as written may allow an ST writer to add components and not require auditing on the functionality provided by these components if the FAU_GEN.1-NIAP-0407 elements are used directly as indicated in the CC. Also, the FAU_GEN.2-NIAP-0410 component should be included as stated in the interpretation.
Therefore, the PPRB recommends the following standard wording and format (including the table) be used when FAU_GEN.1-NIAP-0407 and FAU_GEN.2-NIAP-0410 are included in the PP. The table in FAU_GEN.1-NIAP-0407 is for illustrative purposes only; the PP writing team should detail audit information as required for their PP.

When constructing the table, the PP authors should consider the "Basic" level of audit the starting point for selecting the events to be audited. However, when examining the Basic level of audit for each component included in the PP, the PP authors may choose to either omit or add events. The PP authors should examine other medium robustness PPs to determine in what instances strict adherence to the CC Basic level of audit may not be appropriate.

Required Text

Instruction 16-1: FAU_GEN.1-NIAP-0407 Audit data generation

FAU_GEN.1-NIAP-0407 Audit data generation

FAU_GEN.1.1-NIAP-0407 – The TSF shall be able to generate an audit record of the following auditable events:

- Start-up and shutdown of the audit functions;
- All auditable events listed in Table 1;
- [selection: [assignment: *events at a basic level of audit introduced by the inclusion of additional SFRs determined by the ST author*], [assignment: *events commensurate with a basic level of audit introduced by the inclusion of explicit requirements determined by the ST author*], "*no additional events*"].

Application Note: For the selection, the ST author should choose one or both of the assignments (as detailed in the following paragraphs), or select "no additional events".

For the first assignment, the ST author augments the table (or lists explicitly) the audit events associated with the basic level of audit for any SFRs that the ST author includes that are not included in this PP.

Likewise, for the second assignment the ST author includes audit events that may arise due to the inclusion of any explicit requirements not already in the PP. Because "basic" audit is not defined for such requirements, the ST author will need to determine a set of events that are commensurate with the type of information that is captured at the basic level for similar requirements.

If no additional (CC or explicit) SFRs are included, or if additional SFRs are included that do not have "basic" audit associated with them, then it is acceptable to assign "no additional events" in this item.

FAU_GEN.1.2-NIAP-0407 - The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [information specified in column three of Table 1 below].

Appl*ication Note: In column 3 of the table below, "if applicable" is used to designate data that should be included in the audit record if it "makes sense" in the context of the event that generates the record. If no other information is required (other than that listed in Item a above) for a particular audit event type, then an assignment of "none" is acceptable.*

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1-NIAP-0407	None	
FAU_SAR.1	Opening the audit trail	The identity of the <role administrator> performing the function
FAU_SAR.2	Unsuccessful attempts to read information from the audit records	The identity of the <role administrator>performing the function
FAU_SAR.3	None	
FAU_SEL.1-NIAP-0407	All modifications to the audit configuration that occur while the audit collection functions are operating	The identity of the <role administrator>performing the function
(…all components in the PP should be included in this table…)		

Table 1 – Auditable Events

Instruction 16-2: FAU_GEN.2-NIAP-410 User Identity Association

FAU_GEN.2-NIAP-0410 User identity association

FAU_GEN.2.1-NIAP-0410 For audit events resulting from actions of identified users, the TSF shall be able to associate each auditable event with the identity of the user that caused the event.

Instruction 17: FAU_SEL.1-NIAP-0407 Audit event selection
(Back to TOC)

The following text reflects the consistent selections and assignments that the PPRB recommends for all medium robustness PPs. PP authors should also include other technology-specific attributes on which to base the selectivity of audit.

Required Text

FAU_SEL.1-NIAP-0407 Selective Audit

FAU_SEL.1.1-NIAP-0407 - **Refinement:** The TSF shall **allow only the <role administrator>** to include or exclude auditable events from the set of audited events based on the following attributes:

a) user identity;

b) event type;

c) [selection: *object identity, subject identity, host identity, "none"*];

d) success of auditable security events;

e) failure of auditable security events; and

f) [selection: [assignment: *list of additional criteria that audit selectivity is based upon], no additional criteria*]].

Application Note: "event type" is to be defined by the ST author; the intent is to be able to include or exclude classes of audit events.

Instruction 18: FAU_STG.1-NIAP-0429 Audit event storage
(Back to TOC)

The PPRB recommends that the administrative role allowed to delete audit records be specifically specified in the requirement, and that modifications to the audit records in the audit trail be prevented. In order to implement these changes, as well as the interpretations to the FAU_STG.1 requirement, the following text and format should be used for medium robustness PPs.

Note that I-0423 changes FAU_STG.1.2 from "modifications" to "unauthorized modifications"; the PPRB recommends that *all* modifications (whether authorized or not) be prevented, thus the refinement for FAU_STG.1.2-NIAP-0429 below is suggested.

Required Text:

FAU_STG.1-NIAP-0429 Protected audit trail storage

FAU_STG.1.1-NIAP-0429 – **Refinement:** The TSF shall **restrict the deletion of stored** audit records in the audit trail **to the** <role administrator>.

FAU_STG.1.2-NIAP-0429 **Refinement:** The TSF shall be able to *prevent* modifications to the audit records in the audit trail**.**

Instruction 19: FAU_STG.3 Audit event storage
(Back to TOC)

Should the PP author invoke FAU_STG.3, it should be structured in a common manner to reflect the same assignments across all medium robustness PPs.

This requirement calls for the percentage of the storage capacity to be administrator settable; this implies that an FMT_MOF or FMT_MTD requirement is needed as well. PP Authors should ensure that it is included when this component is included.

Required Text

FAU_STG.3 Action in case of possible audit data loss

FAU_STG.3.1 - **Refinement**: The TSF shall [immediately alert the ***<role administrator>***by displaying a message at the local console, [assignment: other actions determined by the ST author]] if the audit trail exceeds [a ***<role administrator>***-settable percentage of storage capacity].

Application Note: The ST Author should determine if there are other actions that should be taken when the audit trial setting is exceeded, and put these in the assignment. If there are no other actions, then a null assignment is acceptable.

Instruction 20: FAU_STG.NIAP-0414 Site-Configurable Prevention of Audit Loss
(Back to TOC)

The PPRB recommends that the PP author specify functionality for audit trail loss for medium robustness PPs. Since it is desirable that this capability be administrator-settable, FAU_STG.NIAP-0414-1 should be used as follows.

FAU_STG.NIAP-0414-1 calls for the selection of the option taken by the administrator when there's an audit storage failure. The inclusion of requirement in the PP implies that an FMT_MOF or FMT_MTD requirement is needed as well. PP Authors should ensure

that it is included when this component is included. If there are "special" administrators that are able to perform this function, then the application note and the text of the requirement should be changed as well.

Required Text

FAU_STG.NIAP-414 Site-configurable Prevention of audit data loss

FAU_STG.NIAP-0414-1. The TSF shall provide the <role administrator> the capability to select one or more of the following actions [selection: *'ignore auditable events', 'prevent auditable events, except those taken by the authorised user with special rights', 'overwrite the oldest stored audit records'*] and [assignment: *other actions to be taken in case of audit storage failure*] to be taken if the audit trail is full.

FAU_STG.NIAP-0414-2-NIAP-0429 Refinement: The TSF shall **enforce the <role administrator>'s** [*selection: choose one of: "ignore auditable events", "prevent auditable events, except those taken by the authorized user with special rights", "overwrite the oldest stored audit records"*] and [*assignment: other actions to be taken in case of audit storage failure*] if the audit trail is full.

Application Note: The TOE provides the <role administrator> the option of preventing audit data loss by preventing auditable events from occurring. The <role administrator>r's actions under these circumstances are not required to be audited. The TOE also provides the <role administrator> the option of overwriting "old" audit records rather than preventing auditable events, which may protect against a denial-of-service attack.

The ST writer should fill in other technology-specific actions that can be taken for audit storage failure (in addition to the two already specified), or select "no additional options" if there are no such technology-specific actions.

Instruction 21: Security alarm

(Back to TOC)

The PPRB considers a more robust audit mechanism essential to the assurance afforded by medium robustness TOEs. The PPRB suggests using the following requirements to implement this functionality. If remote administration is not feasible for the technology, then the PP authors should consider modifying the following requirements appropriately.

Required Text

Instruction 21-1: FAU_ARP.1 Security alarm

FAU_ARP.1 Security alarms

Dependencies: FAU_SAA.1 Potential violation Analysis

FAU_ARP.1.1 – The TSF shall *[immediately display a message identifying the potential security violation, and make accessible the audit record contents associated with the auditable event(s) that generated the alarm, at the:*

a. *local console;*

b. *remote <administrative role' s> sessions that exist;*

c. *remote <administrative role' s> sessions that are initiated before the alarm has been acknowledged; and*

d. [*selection: [ST assignment: other methods determined by the ST author], no other methods]*]

upon detection of a potential security violation.

Application Note: The TSF provides a message to the local console regardless of whether an administrator is logged in. The message is displayed at the remote console if an administrator is already logged in, or when an administrator logs in if the alarm message has not been acknowledged. The audit records contents associated with the alarm may or may not be part of the message displayed, however the relevant audit information must be available to administrators. In addition, the TOE provides an audible alarm that can be configured to sound an alarm if desired by the Security Administrator. It is acceptable for the ST author to fill the open assignment with none, if no other methods (e.g., pager, e-mail) are included in the TOE.

Instruction 21-2: FAU_ARP_ACK_(EXP).1 Security alarm acknowledgment

Explicit: Security alarm acknowledgement (FAU_ARP_ACK_(EXP).1)

Dependencies: FAU_SAA.1 Potential violation Analysis

FAU_ARP_ACK_(EXP).1.1 – The TSF shall display the alarm message identifying the potential security violation and make accessible the audit record contents associated with the auditable event(s) until it has been acknowledged. An audible alarm will sound until acknowledged by an administrator.

FAU_ARP_ACK_(EXP).1.2 – The TSF shall display an acknowledgement message identifying a reference to the potential security violation, a notice that it has been acknowledged, the time of the acknowledgement and the user identifier that acknowledged the alarm, at the:

- local console, and
- remote administrator sessions that received the alarm.

Application Note: This explicit requirement is necessary since a CC requirement does not exist to ensure an administrator will be aware of the alarm. The intent is to ensure that if an administrator is logged in and not physically at the console or remote workstation the message will remain displayed until they have acknowledged it. The message will not be scrolled off the screen due to other activity-taking place (e.g., the Audit Administrator is running an audit report). If the Security Administrator

configures the TOE to generate an audible alarm, the alarm will sound until an administrator acknowledges the alarm. Acknowledging the message and audible alarm could be a single event, or different events.

FAU_ARP_ACK_(EXP).1.2 ensures that each administrator that received the alarm message also receives the acknowledgement message, which includes some form of reference to the alarm message, who acknowledged the message and when.

Instruction 22: FAU_SAA.1-NIAP-407 Potential violation analysis
(Back to TOC)

The PP authors should consider the unique technology-dependant events that would make sense to include as indicators of a potential violation of the policies being enforced by that specific technology. If remote administration is not feasible for the technology, then the PP authors should consider modifying the following requirements appropriately.

Required Text

FAU_SAA.1-NIAP-0407 Potential violation analysis

FAU_SAA.1.1-NIAP-0407 – The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the TSP.

FAU_SAA.1.2-NIAP-0407 - **Refinement:** The TSF shall **monitor the** accumulation or combination of the following events known to indicate a potential security violation:

a) <role administrator>-specified number of authentication failures;

b) Any detected replay of TSF data or security attributes;

c) Any failure of the cryptographic self-tests;

d) Any failure of the other TSF self-tests;

e) <role administrator>-specified number of encryption failures;

f) <role administrator>-specified number of decryption failures; and

g) [selection: [*assignment: additional events from the set of defined auditable events], "no additional events"*].

Application Note: The intent of this requirement is that an alarm is generated (FAU_ARP.1) once the threshold for an event is met. Once the alarm has been generated it is assumed that the "count" for that event is reset to zero. The <role administrator>-settable number of authentication failures in (a) is intended to be the same value as specified in FIA_AFL.1.

The failure of TSF self-tests in (d) include failures of FPT_TST_(EXP).

B. Cryptographic Support

(Back to TOC)

Instruction 23: Cryptographic Support

(Back to TOC)

The TSF may employ cryptographic functionality to help satisfy several high-level security objectives. These include (but are not limited to): identification and authentication, non-repudiation, trusted path, trusted channel and data separation.

Cryptographic services might be provided in hardware or software, and might be provided at any level from link up through application. Cryptography might be based upon public-keys or on private key exchanges, and is implemented using any of a variety of algorithms, some of which can be certified under validation programs such as the Federal Information Processing Standard (FIPS). Additionally, the cryptographic support requirements of one technology may not be suitable for a different technology. Each technology area has unique requirements that involve a team effort to ensure that all aspects of the technology are covered[10]. Because of all of these factors, there are a considerable number of ways to express FCS components, including refined and extended components, that PP authors might use to express the cryptographic needs unique to the technology area of their PP. This makes it imperative that the TAL collaborate with the Cryptographic Support Organization so all the required cryptographic support requirements suitable for medium robustness are accurately defined relative to the technology area.

As required text

Instruction 23-1: FCS_BCM_(EXP).1 Baseline Cryptographic Module

Baseline Cryptographic Module (FCS_BCM_(EXP).1)

FCS_BCM_(EXP).1.1 All cryptographic modules shall comply with FIPS PUB 140-2 when performing FIPS-approved cryptographic functions in FIPS-approved cryptographic modes of operation.

FCS_BCM_(EXP).1.2 Cryptographic functions and cryptographic modes of operation as identified in this PP shall be NSA-validated.

Application Note: In time, OS PP cryptographic requirements are expected to evolve such that NSA-validated cryptographic modules shall only contain cryptographic functions, cryptographic modes of operation, and other types of cryptographic processing that are compliant with this PP.

FCS_BCM_(EXP).1.3 All cryptographic modules implemented in the TSF ***[selection:***

[10] Cryptography presents a unique challenge in that there are many technologies that perform the cryptography itself; others use a Cryptographic Application Program Interface (CAPI) to another product or the underlining operating system.

- ***Entirely in hardware shall have a minimum overall rating of FIPS PUB 140-2, Level 3;***
- ***Entirely in software shall have a minimum overall rating of FIPS PUB 140-2,Level 1 and also meet FIPS PUB 140-2,Level 3 for the following: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; Cryptographic Key Management; Design Assurance; and FIPS PUB 140-2,Level 4 Self Tests[11] as defined by this PP;***
- ***As a combination of hardware and software shall have a minimum overall rating of FIPS PUB 140-2, Level 1 and also meet FIPS PUB 140-2,Level 3 for the following: Cryptographic Module Ports and Interfaces; Roles, Services and Authentication; Cryptographic Key Management; Design Assurance; and FIPS PUB 140-2,Level 4 Self Tests[12] as defined by this PP.]***

Application Note: "Combination of hardware and software" means that some part of the cryptographic functionality will be implemented as a software component of the TSF. The combination of a cryptographic hardware module and a software device driver whose sole purpose is to communicate with the hardware module is considered a hardware module rather than a "combination of hardware and software".

Instruction 23-2: FCS_CKM Cryptographic Key Management

Cryptographic support requirements suitable for medium robustness are accurately defined relative to the technology area as determined by TAL and Cryptographic Support Organization.

Instruction 23-3: FCS_COP Cryptographic operation

Cryptographic support requirements suitable for medium robustness are accurately defined relative to the technology area as determined by TAL and Cryptographic Support Organization.

C. User Data Protection

(Back to TOC)

[11] Security Level 4 Self Tests comprise the Security Level 1 Self Tests in FIPS PUB 140-2 and the Statistical RNG Tests in Appendix C of this PP. These Statistical RNG Tests are the same as those included in the 25 May 2001 version of FIPS PUB 140-2.

[12] See previous footnote.

Instruction 24: FDP_ACF.1 Access control functions
(Back to TOC)

If the PP authors choose to use the FDP_ACF family requirements, they should use the following interpreted requirement text as a basis.

Interpreted Text:

FDP_ACF.1-NIAP-0407 Security attribute based access control

FDP_ACF.1.1-NIAP-0407: The TSF shall enforce the [assignment: access control SFP] to objects based on the following: [assignment: list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes]

FDP_ACF.1.2-NIAP-0407 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [assignment: *rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects*].

FDP_ACF.1.3-NIAP-0407 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [selection: [assignment: *rules, based on security attributes, that explicitly authorise access of subjects to objects*], "*no additional rules*"].

FDP_ACF.1.4-NIAP-0407 The TSF shall explicitly deny access of subjects to objects based on the [selection: [assignment: *rules, based on security attributes, that explicitly deny access of subjects to objects*], "*no additional rules*"].

Instruction 25: Reserved
(Back to TOC)

D. Identification and Authentication
(Back to TOC)

Instruction 26: FIA_AFL.1Authentication failures
(Back to TOC)

The PPRB recommends that authentication failure controls be present on all medium robustness PPs, and further that these controls be administrator-settable. The PPRB recommends the following text be included to capture this functionality for all medium robustness PPs.

Required Text:

FIA_AFL.1 Authentication failure handling

Dependencies: FIA_UAU.1 Timing of authentication

FIA_AFL.1.1 The TSF shall detect when *a Security Administrator configurable positive integer within [a Security Administrator configurable amount of time]* unsuccessful authentication attempts occur related to [a user's authentication].

FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall [lock the device for a Security Administrator configurable amount of time].

Application Note: At least one account should be exempted from the FIA_AFL.1.2 requirement in order to prevent denial of access.

Interp note: This requirement is modified as per CCIMB Interp #111

The PP authors should ensure that when the ***entities requesting authentication*** is specified in the PP, at least one account should be exempted from the requirement so as to avoid an administrative denial of service.

Instruction 27: FIA_USB.1 User-subject binding
(Back to TOC)

In the *Threats, Policies, Objectives, and Requirements for medium robustness TOEs* table the PPRB suggests including FIA_USB.1 below. This text is included below to capture the notion that all of the user attributes specified in FIA_ATD should be associated with subjects.

Required Text:

FIA_USB.1 User-subject binding

Dependencies: FIA_ATD.1 User attribute definition

FIA_USB.1.1: The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: [assignment: list of user security attributes].

FIA_USB.1.2: The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: [assignment: rules for the initial association of attributes].

FIA_USB.1.3: The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: [assignment: rules for the changing of attributes].

Application Note: User security attributes are defined in FIA_ATD.1

Interp note: This requirement is modified as per CCIMB Interp #137.

E. Protection of the TSF

(Back to TOC)

Instruction 28: FPT_RPL.1 Replay detection

(Back to TOC)

In order to ensure consistency in the selection of data and actions for which replay detection is required at medium robustness, the PPRB recommends that the following text be used.

Required Text

FPT_RPL.1 Replay detection

FPT_RPL.1.1 - The TSF shall detect replay for the following entities: [authentication data, TSF data, and security attributes].

FPT_RPL.1.2 - The TSF shall perform: [reject data; audit event; and [assignment: *list of specific actions*]] when replay is detected.

Instruction 29: FPT_RCV.2 Trusted recovery

(Back to TOC)

The PPRB considers basic recovery a feature consistent with medium robustness. The PPRB suggests including the following text for a minimum of FPT_RCV.2 in all medium robustness PPs. For medium robustness, a selection of "no failures/service discontinuities" is acceptable. However, the PP authors may wish to leave the selection open to accommodate vendors that do provide more robust recovery mechanisms.

The PP authors may choose to use FPT_RCV.3 instead of FPT_RCV.2.

Required Text:

FPT_RCV.2 Automated recovery

FPT_RCV.2.1 When automated recovery from [assignment: *list of failures/service discontinuities*] is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.2.2 For [assignment: *list of failures/service discontinuities*], the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3 Automated recovery without undue loss

Dependencies: FPT_TST.1 TSF testing, AGD_ADM.1 Administrator guidance, ADV_SPM.1 Informal TOE security policy model

Hierarchical to: FPT_RCV.2

FPT_RCV.3.1 When automated recovery from [assignment: *list of failures/service discontinuities*] is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2 For [assignment: *list of failures/service discontinuities*], the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3 The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding [assignment: *quantification*] for loss of TSF data or objects within the TSC.

FPT_RCV.3.4 The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

Instruction 30: FPT_TST TSF Self test
(Back to TOC)

The PPRB recommends that TSF testing be specified in all medium robustness PPs in order to validate aspects of the TSF prior to or while it is operating. However, some TOE data are dynamic (e.g., data in the audit trail, passwords) and so interpretation of "integrity" for FPT_TST.1.2 is required, leading to potential inconsistencies amongst medium robustness TOEs. The PPRB therefore makes the following recommendation for the FPT_TST component.

Required Text:

FPT_TST_(EXP).4 TSF testing (with cryptographic integrity verification)

Dependencies: FPT_AMT.1 Abstract Machine testing

FPT_TST_(EXP).4.1 –The TSF shall run a suite of self-tests during initial start-up, periodically during normal operation as specified by the <role administrator>, and at the request of a <role administrator>

to demonstrate the correct operation of the hardware portions of the TSF.

FPT_TST_(EXP).4.2 –The TSF shall provide a <role administrator> with the capability to use a TSF-provided cryptographic function to verify the integrity of all TSF data except the following: audit data, *[selection: [assignment: other dynamic TSF data for which no integrity validation is justified], none]].*

FPT_TST_(EXP).4.3 - The TSF shall provide a <role administrator> with the capability to use a TSF-provided cryptographic function to verify the integrity of stored TSF executable code.

Application Note: This explicit requirement is necessary since some TOE data are dynamic (e.g., data in the audit trail, passwords) and so interpretation of "integrity" for FPT_TST.1.2 is required, leading to potential inconsistencies. The intention is that any parameter that only an administrator can control is verified to ensure its integrity is maintained. It is not necessary for the TOE to verify the integrity of audit data or user's passwords. If the TOE verifies the integrity of these, the ST author may fill in the assignment to include them.

Since this TOE includes all the hardware necessary for the operation of the TOE, the element FPT_TST_(EXP).4.1 ensures that the hardware aspects of the TOE are tested prior to or during operations. It is not necessary to test the software portions of the TSF, since the evaluation ensures the correct operation of the software, software does not degrade or suffer intermittent faults, as does hardware, and integrity of the software portions of the TSF are addressed by FPT_TST_(EXP).4.3. Note that since cryptographic functions implemented in hardware that are part of a cryptomodule are tested in FPT_TST_(EXP).5, this requirement only applies to cryptographic functionality implemented in hardware that is not implemented in a cryptomodule (for instance, an implementation of a Key Agreement algorithm).

In element 4.2, the ST author should specify the TSF data for which integrity validation is not required, and also specify the administrative role that is able to invoke the integrity verification process. While some TSF data are dynamic and therefore not amenable to integrity verification, it is expected that all TSF data for which integrity verification "makes sense" be subject to this requirement.

In elements 4.2 and 4.3, the cryptographic mechanism can be any one of the ones specified in FCS_COP_(EXP).3 or FCS_COP_(EXP).6, although typically hash functions or digital signatures are used for integrity verification.

FPT_TST_(EXP).5 Cryptographic self-test

Dependencies: FPT_AMT.1 Abstract Machine testing

FPT_TST_(EXP).5.1 – The TSF shall run the suite of self-tests provided by the FIPS 140-2 cryptographic module during initial start-up (power on), at the request of the cryptographic administrator, periodically (at a Security Administrator-specified interval not less than at least

once a day) to demonstrate the correct operation of the cryptographic components of the TSF.

FPT_TST_(EXP).5.2 – The TSF shall be able to run the suite of self-tests provided by the FIPS 140-2 cryptographic module immediately after the generation of a key.

Application Note: For element 3.2, the Cryptographic Administrator has the ability to enable and disable this capability; this is specified in FMT_MOF.1(2).

Instruction 31: FPT_AMT.1 Abstract Machine Testing
(Back to TOC)

Required Text

FPT_AMT.1 Abstract Machine Testing

FPT_AMT.1.1 **Refinement:** The TSF shall run a suite of tests during the initial start-up and also periodically during normal operation, or at the request of an authorized **administrator** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the **software portions of the** TSF.

Application Note: The test suite need only cover aspects of the underlying abstract machine on which the TSF relies to implement required functions, including domain separation.

F. Resource Utilization

(Back to TOC)

Instruction 32: Resource Utilization/Management

(Back to TOC)

Another key feature of medium robustness TOEs is their ability to prevent some level of denial of service attacks. These types of attacks are very technology-specific and must be specified by the PP authors. It is not necessary for a medium robustness TOE to counter all denial of service attacks; only those that may be countered using current technology capabilities should be specified.

In specifying requirements for resource utilization, the PP authors need to use three different components. Because there are a number of selections of these components, it may be necessary to iterate them to distinguish requirements on one resource from another. FRU_RSA.1 should be used to specify the resource, and controls on that resource. The PPRB suggests refining the requirement to specify that the administrator be able to specify (at a minimum) the period of time over which the resource utilization check will be made; therefore, an FMT_MOF.1 iteration is needed to restrict this functionality to the administrator. Likewise, since the administrator is imposing limits on the use of a resource, FMT_MTD.2 iteration is needed to specify those limits and restrict them to the appropriate administrator.

The following text is an example of such a FRU_RSA.1/FMT_MOF.1/FMT_MTD.2 grouping from the Firewall PP. Unlike other requirement text in this document, it *is not* intended to be used verbatim in other PPs. Instead, it is included as an example of how the three components are linked to specify this type of functionality, and to suggest a style (including the somewhat verbose application notes) for such components. The PPRB suggests that the PP authors use non-technology-specific refinements (such as those mandating an administrator be allowed to set various items called for by the components, as opposed to having the developer "hard-wire" them in) in their specifications.

Example Text

Instruction 32-1: FRU_RSA.1 Resource allocation

> **FRU_RSA.1(2) - Maximum quotas (controlled connection-oriented quotas)**
>
> FRU_RSA.1.1(2) – **Refinement:** The TSF shall enforce **administrator-specified** maximum quotas of the following resources: [assignment: controlled connection-oriented resources] that ***users associated with*** [*an administrator-specified network identifier and a set of administrator-specified network identifiers*] can use [over an ***administrator-****specified period of time]*.

Application Note: This requirement applies to a network entity attempting to exhaust the specified connection-oriented resources (or set of such resources) on the TOE. Connectionless sessions are not a concern because they do not consume resources that persist like connection-oriented sessions do.

The ST author should fill in the first assignment with the list of connection-oriented resources to which this requirement applies. That is, when a network entity uses such a connection-oriented resource (or a collection of these resources), the TOE tracks that use for the purpose of determining whether the entity has exceed the quota established by the administrator.

The ST author should use the first selection to indicate whether the TOE is able to track the assignment of the specified resources based on a single network identifier (e.g., a specific IP address) or multiple network identifiers (e.g., a specific IP subnet address). The second selection should reflect the way in which the TOE tracks such resource use. Note that the ST author may have to iterate this requirement if different resources can be controlled differently by the TOE. The ST author should ensure that FMT_MTD.2(2) specifies the actions that are taken for each resource on which there is a quota.

Instruction 32-2: FMT_MOF.1 Management of functions in TSF

FMT_MOF.1(4) Management of security functions behavior (quota mechanism)

FMT_MOF.1.1(4) - The TSF shall restrict the ability to *determine the behavior of* the functions:

1 [Controlled connection-oriented resource allocation (FRU_RSA.1(2));

2 an administrator-specified network identifier;

3 set of administrator-specified network identifiers;

4 administrator-specified period of time.]

to [the Security Administrator].

Application Note: "determine the behavior of" refers to specifying the network identifier(s) that will be tracked using the FRU_RSA.1(2) requirement and the time period over which the quota limitations are enforced. Note that the specification of the actual quotas, while part of the resource allocation functionality, is done by FMT_MTD.2(2).

Instruction 32-3: FMT_MTD.2 Management of TSF data

FMT_MTD.2(2) Management of limits on TSF data (controlled connection-oriented quotas)

FMT_MTD.2.1(2) - The TSF shall restrict the specification of the limits for [quotas on controlled connection-oriented resources] to [the Security Administrator].

FMT_MTD.2.2(2) - The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: [assignment: actions to be taken].

Application Note: For FMT_MTD.2.2(2), the ST author should specify the actions that the TOE takes for each controlled connection-oriented resource when the quota (with respect to the specific network identifier or set of network identifiers) established by the Security Administrator is reached. This requirement may have to be iterated to be consistent with FRU_RSA.1(2). See the application note on FRU_RSA.1(2) for more detail on the requirements for the quota mechanism.

G. Security Management Roles

(Back to TOC)

Instruction 33: FMT_SMR.2 Restriction on Security Roles

(Back to TOC)

Separation of roles is required in medium robustness PPs primarily in order to mitigate the T.ADMIN_ROGUE threat. Additionally, the PPRB considers remote administration desirable for medium robustness TOEs. However, remote administration may not make sense for all technology areas. If remote administration does not make sense, the PP authors should provide a justification for this -- separate from the PP--and adjust the text of the PP appropriately for the P.ADMIN_ACCESS policy. The PP authors should also modify the appropriate threats, policies, objectives, and components to remove the notion of remote administration. When remote administration is employed the TOE must provide a secure means of performing the remote administration by providing a means for protecting the communication path from disclosure of data, and providing a means for detecting modification of data.

Required Text

FMT_SMR.2 Restrictions on security roles

FMT_SMR.2.1 - The TSF shall maintain the roles:

[Security Administrator; Cryptographic Administrator (i.e., users authorized to perform cryptographic initialization and management functions); Audit Administrator; and [*selection: [assignment: any other roles], none*]].

FMT_SMR.2.2 - The TSF shall be able to associate users with roles.

FMT_SMR.2.3 - The TSF shall ensure that the conditions

[All roles shall be able to administer the TOE locally; all roles shall be able to administer the TOE remotely; all roles are distinct; that is, there shall be no overlap of operations performed by each role, with the following exceptions: [assignment: the PP author assigns the functions that are allowed to overlap] (*The PP author must play close attention to the FMT requirements and which roles are allowed to perform certain **functions** within the other requirements.)*.

Application Note: The administering of the TOE is limited to the capabilities associated with an administrative role.

H. TOE Access

(Back to TOC)

Instruction 34: FTA_TAB.1 TOE access banner

(Back to TOC)

The PPRB recommends that a TOE Access Banner be required for all medium robustness TOEs. The PP authors should ensure that the wording of the requirement reflects -the fact that a banner only makes sense for sessions established by human users. Note also that the application note clarifies that an administrator has control of what is displayed, including whether or not to display information that might identify the TOE (as opposed to the developer "hard-coding" this information).

Required Text

FTA_TAB.1 Default TOE access banners

FTA_TAB.1.1 - **Refinement:** Before establishing a user session **that requires authentication**, the TSF shall display **only** a **<role administrator>-specified** advisory **notice and consent** warning message regarding unauthorized use of the TOE.

Application Note: The access banner applies whenever the TOE will provide a prompt for identification and authentication (e.g., administrators, authenticated proxy users). The intent of this requirement is to advise users of warnings regarding the unauthorized use of the TOE and to provide the Security Administrator with control over what is displayed (e.g., if the Security Administrator chooses, they can remove banner information that informs the user of the product and version number).

Instruction 35: FTA_TSE.1 TOE session establishment

(Back to TOC)

The PPRB recommends that additional restrictions be placed on how and when authorized users can access the TOE; this is accomplished by FTA_TSE.1. In order to ensure a similar granularity of control with this mechanism, the PPRB recommends the following text be used in the assignment for medium robustness PPs (**location, time, and day)**. PP authors may have to include an application note to clarify what is meant by "authorized user session."

Required Text

FTA_TSE.1 TOE session establishment

Hierarchical to: No other components.

Dependencies: No dependencies

FTA_TSE.1.1 The TSF shall be able to deny session establishment based on [assignment: *attributes*].

V. Explicit CC Security Assurance Requirements

(Back to TOC)

Instruction 36: Explicit Assurance Requirements

The PPRB has crafted a number of explicit assurance requirements to be included in profiles written for medium robustness environments. The assurance requirements are provided below. The PP author should place the assurance requirements in the body of the PP. Additional explanatory (e.g., objective, application notes) material (as provided) for the explicit component can be found in an appendix of this document and should be incorporated (as an appendix) into medium robustness PPs as well. The ADV_INT_(EXP).1 assurance requirement differs from other assurance requirements in that the PP author is to fill in an assignment of the modules that are of special concern to their TOE.

Required text

Instruction 36-1: ADV_ARC_(EXP).1 Architectural design

ADV_ARC_(EXP).1 Architectural design

Dependencies: FPT_SEP.1, FPT_RVM.1, ADV_FSP_(EXP).1, ADV_HLD_(EXP).1, ADV_LLD_(EXP).1, ADV_INT_(EXP).1, ADV_IMP.2

ADV_ARC_(EXP).1.1D The developer shall provide the architectural design of the TSF.

Content and Presentation of Evidence:

ADV_ARC_(EXP).1.1C The presentation of the architectural design of the TSF shall be informal.

ADV_ARC_(EXP).1.2C The architectural design shall be internally consistent.

ADV_ARC_(EXP).1.3C The architectural design shall describe the design of the TSF self-protection mechanisms.

ADV_ARC_(EXP).1.4C The architectural design shall describe the design of the TSF in detail sufficient to determine that the security enforcing mechanisms cannot be bypassed.

ADV_ARC_(EXP).1.5C The architectural design shall justify that the design of the TSF achieves the self-protection function.

Evaluator action elements:

ADV_ARC_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_ARC_(EXP).1.2E The evaluator shall analyze the architectural design and dependent documentation to determine that FPT_SEP and FPT_RVM are accurately implemented in the TSF.

Instruction 36-2: ADV_INT_(EXP).1 Modular decomposition

ADV_INT_(EXP).1 Modular decomposition

Dependencies: ADV_IMP.2, ADV_LLD_(EXP).1

Developer action elements:

ADV_INT_(EXP).1.1D The developer shall design and implement the TSF using modular decomposition.

ADV_INT_(EXP).1.2D The developer shall use sound software engineering principles to achieve the modular decomposition of the TSF.

ADV_INT_(EXP).1.3D The developer shall design the modules such that they exhibit good internal structure and are not overly complex.

ADV_INT_(EXP).1.4D The developer shall design modules that implement the **[assignment: list of SFPs]** such that they exhibit only functional, sequential, communicational, or temporal cohesion, with limited exceptions.

ADV_INT_(EXP).1.5D The developer shall design the SFP-enforcing modules such that they exhibit only call or common coupling, with limited exceptions.

Application Note: SFP-enforcing modules are TSF modules that implement a specific SFP identified in ADV_INT_(EXP).1.4D.

ADV_INT_(EXP).1.6D The developer shall implement TSF modules using coding standards that result in good internal structure that is not overly complex.

ADV_INT_(EXP).1.7D The developer shall provide a software architectural description.

Content and presentation of evidence elements:

ADV_INT_(EXP).1.1C The software architectural description shall identify the SFP-enforcing and non-SFP-enforcing modules.

ADV_INT_(EXP).1.2C The TSF modules shall be identical to those described by the low level design (ADV_LLD_(EXP).1.4C).

ADV_INT_(EXP).1.3C The software architectural description shall provide a justification for the designation of non-SFP-enforcing modules that interact with the SFP-enforcing module(s).

ADV_INT_(EXP).1.4C The software architectural description shall describe the process used for modular decomposition.

ADV_INT_(EXP).1.5C The software architectural description shall describe how the TSF design is a reflection of the modular decomposition process.

ADV_INT_(EXP).1.6C The software architectural description shall include the coding standards used in the development of the TSF.

ADV_INT_(EXP).1.7C The software architectural description shall provide a justification, on a per module basis, of any deviations from the coding standards.

ADV_INT_(EXP).1.8C The software architectural description shall include a coupling analysis that describes inter-module coupling for the SFP-enforcing modules.

ADV_INT_(EXP).1.9C The software architectural description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by SFP-enforcing modules, other than those permitted.

ADV_INT_(EXP).1.10C The software architectural description shall provide a justification, on a per module basis, that the SFP-enforcing modules are not overly complex.

Evaluator action elements:

ADV_INT_(EXP).1.1E The evaluator shall confirm that the information provided meets all the requirements for content and presentation of evidence.

ADV_INT_(EXP).1.2E The evaluator shall perform a cohesion analysis for the modules that substantiates the type of cohesion claimed for a subset of SFP-enforcing modules.

ADV_INT_(EXP).1.3E The evaluator shall perform a complexity analysis for a subset of TSF modules.

Instruction 36-3: ADV_FSP_(EXP).1 Functional specification With Complete Summary

ADV_FSP_(EXP).1 Functional specification with Complete Summary

Dependencies: ADV_RCR.1

Developer Action Elements

ADV_FSP_(EXP).1.1D The developer shall provide a functional specification.

Content and Presentation of Evidence:

ADV_FSP_(EXP).1.1C The functional specification shall completely represent the TSF.

ADV_FSP_(EXP).1.2C The functional specification shall be internally consistent.

ADV_FSP_(EXP).1.3C The functional specification shall describe the external TSF interfaces (TSFIs) using an informal style.

ADV_FSP_(EXP).1.4C The functional specification shall designate each external TSFI as security enforcing or security supporting.

ADV_FSP_(EXP).1.5C The functional specification shall describe the purpose and method of use for each external TSFI.

ADV_FSP_(EXP).1.6C The functional specification shall identify and describe all parameters associated with each external TSFI.

ADV_FSP_(EXP).1.7C For security enforcing external TSFIs, the functional specification shall describe the security enforcing effects and security enforcing exceptions.

ADV_FSP_(EXP).1.8C For security enforcing external TSFIs, the functional specification shall describe direct error messages resulting from security enforcing effects and exceptions.

Evaluator Action Elements

ADV_FSP_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP_(EXP).1.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the user-visible TOE security functional requirements.

Instruction 36-4: ADV_HLD_(EXP).1 Security-enforcing high-level design

ADV_HLD_(EXP).1 Security-enforcing high-level design

Dependencies: FPT_SEP.1, FPT_RVM.1, ADV_FSP_(EXP).1, ADV_LLD_(EXP).1, ADV_ARC_(EXP).1, ADV_INT_(EXP).1

Developer Action Elements:

ADV_HLD_(EXP).1.1D The developer shall provide the high-level design of the TOE.

Content and Presentation of Evidence:

ADV_HLD_(EXP).1.1C The high-level design shall describe the structure of the TOE in terms of subsystems.

ADV_HLD_(EXP).1.2C The high-level design shall be internally consistent.

ADV_HLD_(EXP).1.3C The high level design shall describe the subsystems using an informal style.

ADV_HLD_(EXP).1.4C The high-level design shall describe the design of the TOE in sufficient detail to determine what subsystems of the TOE are part of the TSF.

ADV_HLD_(EXP).1.5C The high-level design shall identify all subsystems in the TSF, and designate them as either security enforcing or security supporting.

ADV_HLD_(EXP).1.6C The high-level design shall describe the structure of the security-enforcing subsystems.

ADV_HLD_(EXP).1.7C For security-enforcing subsystems, the high-level design shall describe the design of the security-enforcing behavior.

ADV_HLD_(EXP).1.8C For security-enforcing subsystems, the high-level design shall summarize any non-security-enforcing behavior.

ADV_HLD_(EXP).1.9C The high-level design shall summarize the behavior for security-supporting subsystems.

ADV_HLD_(EXP).1.10C The high-level design shall summarize all other interactions between subsystems of the TSF.

ADV_HLD_(EXP).1.11C The high-level design shall describe any interactions between the security-enforcing subsystems of the TSF.

ADV_HLD_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD_(EXP).1.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of all user-visible TOE security functional requirements with the exception of FPT_SEP and FPT_RVM.

Instruction 36-5: ADV_LLD_(EXP).1 Security-enforcing low-level design

ADV_LLD_(EXP).1 Security-enforcing low-level design

Dependencies: ADV_FSP_(EXP).1, ADV_HLD_(EXP).1, ADV_ARC_(EXP).1, ADV_IMP.2

ADV_LLD_(EXP).1.1D The developer shall provide the low-level design of the TSF.

Content and Presentation of Evidence

ADV_LLD_(EXP).1.1C The presentation of the low-level design shall be informal.

ADV_LLD_(EXP).1.2C The presentation of the low-level design shall be separate from the implementation representation.

ADV_LLD_(EXP).1.3C The low-level design shall be internally consistent.

ADV_LLD_(EXP).1.4C The low-level design shall identify and describe data that are common to more than one module, where any of the modules is a security-enforcing module.

ADV_LLD_(EXP).1.5C The low-level design shall describe the TSF in terms of modules, designating each module as either security-enforcing or security-supporting.

ADV_LLD_(EXP).1.6C The low level design shall describe each security-enforcing module in terms of its purpose, interfaces, return values from those interfaces, called interfaces to other modules, and global variables.

ADV_LLD_(EXP).1.7C For each security-enforcing module, the low level design shall provide an algorithmic description detailed enough to represent the TSF implementation.

Application Note: An algorithmic description contains sufficient detail such that two different programmers would produce functionally-equivalent code, although data structures, programming methods, etc. may differ.

ADV_LLD_(EXP).1.8C The low level design shall describe each security-supporting module in terms of its purpose and interaction with other modules.

Evaluator Action Elements

ADV_LLD_(EXP).1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD_(EXP).1.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of all TOE security functional requirements, with the exception of FPT_SEP and FPT_RVM.

Instruction 36-6: AVA_CCA_(EXP).2 Systematic Cryptographic Module Covert Channel Analysis

AVA_CCA_(EXP).2 Systematic Cryptographic Module Covert Channel Analysis

Dependencies: ADV_FSP_(EXP).1, ADV_IMP.2, AGD_ADM.1, AGD_USR.1

Application Note: The covert channel analysis is performed only upon the cryptographic module; a search is made for the leakage of critical cryptographic security parameters from the cryptographic module, rather than a violation of an information control policy. Inappropriate handling / leakage of any critical cryptographic security parameters (covered or not) that by design and implementation lie outside the cryptographic module is not addressed by this CCA. Thus, leakage of such parameters in such designs and implementations must be investigated by other means.

AVA_CCA_(EXP).2.1D For the cryptographic module, the developer shall conduct a search for covert channels for the leakage of critical cryptographic security parameters whose disclosure would compromise the security provided by the module.

Application Note: The remainder of the TOE need not be subjected to a covert channel analysis. (Ideally, a covert channel analysis on the entire TSF would determine if TSF interfaces can be used covertly for the leakage of critical cryptographic security parameters. While such extensive covert channel analysis is more complete, it is also difficult and expensive. At this time it is considered beyond the scope of effort and cost considered reasonable for COTS medium robustness products. Consequently, covert channel analysis has been limited here to the cryptographic module, but that analysis limitation does come with some added risk of unknown leakage from other parts of the TOE.)

AVA_CCA_(EXP).2.2D The developer shall provide covert channel analysis documentation.

AVA_CCA_(EXP).2.1C The analysis documentation shall identify covert channels in the cryptographic module and estimate their capacity.

AVA_CCA_(EXP).2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels in the cryptographic module, and the information needed to carry out the covert channel analysis.

AVA_CCA_(EXP).2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA_(EXP).2.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst-case scenarios.

AVA_CCA_(EXP).2.5C The analysis documentation shall describe the worst-case exploitation scenario for each identified covert channel.

AVA_CCA_(EXP).2.6C The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.

AVA_CCA_(EXP).2.1E The NSA evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_CCA_(EXP).2.2E The NSA evaluator shall confirm that the results of the covert channel analysis show that the cryptographic module meets its functional requirements.

AVA_CCA_(EXP).2.3E The NSA evaluator shall selectively validate the covert channel analysis through independent analysis and testing.

Application Note: The cryptographic security parameters are to be defined in the Security Target

VI. Appendices

(Back to TOC)

Appendix A Mapping of Medium Robustness Threats/Policies to Objectives

(Back to TOC)

Sample rationale is provided below. The PP authors should examine various NIAP evaluated PPs for examples of rationale.

Threat/Policy	Objectives Addressing the Threat	Rationale
T. ADMIN_ ERROR An administrator may incorrectly install or configure the TOE, or install a corrupted TOE resulting in ineffective security mechanisms.	O. ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	O. ROBUST_ADMIN_GUIDANCE (ADO_DEL.2, ADO_IGS.1, AGD_ADM.1, AGD_USR.1, AVA_MSU.2) help to mitigate this threat by ensuring the TOE administrators have guidance that instructs them how to administer the TOE in a secure manner and to provide the administrator with instructions to ensure the TOE was not corrupted during the delivery process. Having this guidance helps to reduce the mistakes that an administrator might make that could cause the TOE to be configured in a way that is insecure.
	O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	O.ADMIN_ROLE (FMT_SMR.2) plays a role in mitigating this threat by limiting the functions an administrator can perform in a given role. For example, the Audit Administrator could not make a configuration mistake that would impact the directory access control policy. Likewise, a directory manager could only affect policies in the sub-hierarchy they are responsible for, and not other sub-hierarchies or global directory policies.
	O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	O.MANAGE (FMT_MTD.1) also contributes to mitigating this threat by providing administrators the capability to view configuration settings. For example, if the Security Administrator made a mistake when configuring the rule-set, providing them the capability to view the rules affords them the ability to review the rules and discover any mistakes that might have been made.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.ADMIN_ROGUE An administrator's intentions may become malicious resulting in user or TSF data being compromised.	O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	O.ADMIN_ROLE (FMT_SMR.2) mitigates this threat by restricting the functions available to an administrator. This is somewhat different than the part this objective plays in countering T.ADMIN_ERROR, in that this presumes that separate individuals will be assigned separate roles. If the Audit Administrator's intentions become malicious they would not be able to render the TOE unable to enforce its directory access control policy. On the other hand, if the Directory Administrator becomes malicious they could affect the directory access control policy, but the Audit Administrator may be able to detect those actions.
T.AUDIT_COMPROMISE A malicious user or process may view audit records, cause audit records to be lost or modified, or prevent future audit records from being recorded, thus masking a user's action.	O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information.	O.AUDIT_PROTECTION (FAU.SAR.2, FAU_STG.1-NIAP-0429, FAU_STG.3, FAU_STG.NIAP-0414-1, FMT_SMF.1) contributes to mitigating this threat by controlling access to the audit trail. The auditor and any trusted IT entities performing IDS-like functions are the only ones allowed to read the audit trail. No one is allowed to modify audit records, and the Auditor is the only one allowed to delete audit records in the audit trail. The TOE has the capability to prevent auditable actions from occurring if the audit trail is full, and of notifying an administrator if the audit trail is approaching its capacity. In addition, the TOE has the capability to restore audit data corrupted by the attacker.
	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	O.RESIDUAL_INFORMATION (FDP.RIP.2) prevents a user not authorized to read the audit trail from access to audit information that might otherwise be persistent in a TOE resource (e.g., memory). By ensuring the TOE prevents residual information in a resource, audit information will not become available to any user or process except those explicitly authorized for that data.

Threat/Policy	Objectives Addressing the Threat	Rationale
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the audit trail. Likewise, ensuring that the functions that protect the audit trail are always invoked is also critical to the mitigation of this threat.
T.CRYPTO_COMPROMISE A malicious user or process may cause key, data or executable code associated with the cryptographic functionality to be inappropriately accessed (viewed, modified, or deleted), thus compromising the cryptographic mechanisms and the data protected by those mechanisms.	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	O.RESIDUAL_INFORMATION (FDP_RIP.2) is necessary to mitigate this threat by ensuring no TSF data remain in resources allocated to a user. Even if the security mechanisms do not allow a user to explicitly view TSF data, if TSF data were to inappropriately reside in a resource that was made available to a user, that user would be able to inappropriately view the TSF data.
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering, or unauthorized disclosure.	O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) contributes to countering this threat by ensuring that the TSF can protect itself from users. If the TSF could not maintain and control its domain of execution, it could not be trusted to control access to the resources under its control, which includes the cryptographic data and executable code.
	O.DOCUMENT_KEY_LEAKAGE The bandwidth of channels that can be used to compromise key material shall be documented.	O.DOCUMENT_KEY_LEAKAGE (AVA_CCA_(EXP).2) addresses this threat by requiring the developer to perform an analysis that documents the amount of key information that can be leaked via a covert channel. This provides information that identifies how much material could be inappropriately obtained within a specified time period.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.MASQUERADE A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain access to data or TOE resources.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	O. ROBUST_TOE_ACCESS (FIA_AFL.1, FIA_ATD.1, FIA_UID.2, FIA_UAU.1, FIA_UAU.2, FIA_UAU.5, FTA_TSE.1, AVA_SOF.1, FPT_TDC.1, FPT_ITA.1) mitigates this threat by controlling the logical access to the TOE and its resources. By constraining how and when authorized users can access the TOE, and by mandating the type and strength of the authentication mechanisms, this objective helps mitigate the possibility of a user attempting to login and masquerade as an authorized user. In addition, this objective provides the administrator the means to control the number of failed login attempts a user can generate before an account is locked out, further reducing the possibility of a user gaining unauthorized access to the TOE. This objective also allows the TOE to correctly interpret information used during the authentication process so that it can make the correct decisions when identifying and authenticating users. Finally, this objective provides the ability to control access to certificates and revocation lists so they are available in a timely fashion, contributing to correct authentication decisions.
T.FLAWED_DESIGN Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious user or program.	O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	O.CHANGE_MANAGEMENT (ACM_AUT.1, ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1) plays a role in countering this threat by requiring the developer to provide control of the changes made to the TOE's design. This includes controlling physical access to the TOE's development area, and having an automated configuration management system that ensures changes made to the TOE go through an approval process and only those persons that are authorized can make changes to the TOE's design and its documentation.

Threat/Policy	Objectives Addressing the Threat	Rationale
	O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented..	O.SOUND_DESIGN (ADV_FSP_(EXP).1, ADV_HLD_(EXP).1, ADV_INT_(EXP).1, ADV_LLD_(EXP).1, ADV_ARC_(EXP).1, ADV_RCR.1, ADV_SPM.1) counters this threat, to a degree, by requiring that the TOE be developed using sound engineering principles. By accurately and completely documenting the design of the security mechanisms in the TOE, including a security model, the design of the TOE can be better understood, which increases the chances that design errors will be discovered.
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) ensures that the design of the TOE is independently analyzed for design flaws. Having an independent party perform the assessment ensures an objective approach is taken and may find errors in the design that would be left undiscovered by developers that have a preconceived incorrect understanding of the TOE's design.
T.FLAWED_IMPLEMENTATION Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious user or program.	O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	O.CHANGE_MANAGEMENT (ACM_CAP.4, ACM_SCP.2, ALC_DVS.1, ALC_FLR.2, ALC_LCD.1, ACM_AUT.1) This objective plays a role in mitigating this threat in the same way that the flawed design threat is mitigated. By controlling who has access to the TOE's implementation representation and ensuring that changes to the implementation are analyzed and made in a controlled manner, the threat of intentional or unintentional errors being introduced into the implementation are reduced.

Threat/Policy	Objectives Addressing the Threat	Rationale
	O.SOUND_IMPLEMENTATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented.	In addition to documenting the design so that implementers have a thorough understanding of the design, O.SOUND_IMPLEMENTATION (ADV_IMP.2, ADV_LLD_(EXP).1, ADV_RCR.1, ADV_INT_(EXP).1, ADV_ARC_(EXP).1, ALC_TAT.1) requires that the developer's tools and techniques for implementing the design are documented. Having accurate and complete documentation, and having the appropriate tools and procedures in the development process helps reduce the likelihood of unintentional errors being introduced into the implementation.
	O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	Although the previous three objectives help minimize the introduction of errors into the implementation, O.THOROUGH_FUNCTIONAL_TESTING (ATE_COV.2, ATE_FUN.1, ATE_DPT.2, ATE_IND.2) increases the likelihood that any errors that do exist in the implementation (with respect to the functional specification, high level, and low-level design) will be discovered through testing.
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) helps reduce errors in the implementation that may not be discovered during functional testing. Ambiguous design documentation, and the fact that exhaustive testing of the external interfaces is not required may leave bugs in the implementation undiscovered in functional testing. Having an independent party perform a vulnerability analysis and conduct testing outside the scope of functional testing increases the likelihood of finding errors.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.POOR_TEST Lack of or insufficient tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered thereby causing potential security vulnerabilities.	O.CORRECT_ TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	While these testing activities are necessary for successful completion of an evaluation, this testing activity does not address the concern that the TOE continues to operate correctly and enforce its security policies once it has been fielded. Some level of testing must be available to end users to ensure the TOE's security mechanisms continue to operate correctly once the TOE is fielded. O.CORRECT_ TSF_OPERATION (FPT_TST_(EXP).4, FPT_TST_(EXP).5) ensures that once the TOE is installed at a customer's location, the capability exists that the integrity of the TSF (hardware and software, including the cryptographic functions) can be demonstrated, and thus providing end users the confidence that the TOE's security policies continue to be enforced.
	O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	Design analysis determines that TOE's documented design satisfies the security functional requirements. In order to ensure the TOE's design is correctly realized in its implementation, the appropriate level of functional testing of the TOE's security mechanisms must be performed during the evaluation of the TOE. O.THOROUGH_FUNCTIONAL_ TESTING (ATE_FUN.1, ATE_COV.2, ATE_DPT.2, ATE_IND.2) ensures that adequate functional testing is performed to demonstrate the TSF satisfies the security functional requirements and that the TOE's security mechanisms operate as documented. While functional testing serves an important purpose, it does not ensure the TSFI cannot be used in unintended ways to circumvent the TOE's security policies.
	O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) addresses this concern by requiring a vulnerability analysis be performed in conjunction with testing that goes beyond functional testing. This objective provides a measure of confidence that the TOE does not contain security flaws that may not be identified through functional testing.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.REPLAY A user may gain inappropriate access to the TOE by replaying authentication information, or may cause the TOE to be inappropriately configured by replaying TSF data or security attributes (e.g., captured as transmitted during the course of legitimate use).	O.REPLAY_DETECTION The TOE will provide a means to detect and reject the replay of authentication data as well as other TSF data and security attributes.	O.REPLAY_DETECTION (FPT_RPL.1) prevents a user from replaying authentication data. Prevention of replay of authentication data will counter the threat that a user will be able to record an authentication session between a trusted entity (administrative user or trusted IT entity) and then replay it to gain access to the TOE, as well as counter the ability of a user to act as another user.
T.RESIDUAL_DATA A user or process may gain unauthorized access to data through reallocation of TOE resources from one user or process to another.	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	O.RESIDUAL_INFORMATION (FDP_RIP.2) counters this threat by ensuring that TSF data and user data is not persistent when resources are released by one user/process and allocated to another user/process. This means that network packets sent in response to a request will not have residual data from another packet (potentially from another user) due to the padding of a packet.
T.RESOURCE_EXHAUSTION A malicious process or user may block others from system resources (e.g., *example of resources that apply to technology*) via a resource exhaustion denial of service attack.	O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust <*specific types of resources, which the TOE protects*> resources provided by the TOE (e.g., *examples of resources that apply to technology*).	O.RESOURCE_SHARING (FRU_RSA.1, FMT_MTD.2) mitigates this threat by requiring the TOE to provide controls relating to two different resources: CPU time and available network connections. The administrator is allowed to specify a percentage of processor time that is allowed to be used so that an attempt to exhaust the resource will fail when it reaches the quota. This objective also addresses the denial-of-service attack of a user attempting to exhaust the connection-oriented resources by generating a large number of half-open connections (e.g., SYN attack).
T.SPOOFING A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	It is possible for an entity other than the TOE (a subject on the TOE, or another IT entity on the network between the TOE and the end user) to provide an environment that may lead a user to mistakenly believe they are interacting with the TOE, thereby fooling the user into divulging identification and authentication information. O.TRUSTED_PATH (FTP_ITC.1, FTP_TRP.1) mitigates this threat by ensuring users have the capability to ensure they are communicating with the TOE when providing identification and authentication data to the TOE.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.MALICIOUS_TSF_ COMPROMISE A malicious user or process may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	O.RESIDUAL_INFORMATION (FDP_RIP.2) is necessary to mitigate this threat by ensuring no TSF data remain in resources allocated to a user. Even if the security mechanisms do not allow a user to explicitly view TSF data, if TSF data were to inappropriately reside in a resource that was made available to a user, that user would be able to inappropriately view the TSF data.
	O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	O.SELF_PROTECTION (FPT_SEP.2, FPT_RVM.1) requires that the TSF be able to protect itself from tampering and that the security mechanisms in the TSF cannot be bypassed. Without this objective, there could be no assurance that users could not view or modify TSF data or TSF executables.
	O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	O.MANAGE (FMT_MTD.1, FMT_MSA.1, FMT_MOF.1, FMT_MTD.2, FMT_SMF.1) provides the capability to restrict access to TSF to those that are authorized to use the functions. Satisfaction of this objective (and its associated requirements) prevents unauthorized access to TSF functions and data through the administrative mechanisms.
	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	O.DISPLAY_BANNER (FTA_TAB.1) helps mitigate this threat by providing the Platform Administrator the ability to remove product information (e.g., product name, version number) from a banner that is displayed to users. Having product information about the TOE provides an attacker with information that may increase their ability to compromise the TOE

Threat/Policy	Objectives Addressing the Threat	Rationale
	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data..	O.TRUSTED_PATH (FTP_TRP.1, FTP_ITC.1) plays a role in addressing this threat by ensuring that there is a trusted communication path between the TSF and various users (remote administrators, relying parties (for authentication) and trusted IT entities (for performing replication, for instance)). This ensures the transmitted data cannot be compromised or disclosed during the duration of the trusted path. The protection offered by this objective is limited to TSF data, including authentication data and all data sent or received by trusted IT entities (a relying party's user data is not protected; only the authentication portion of the session is protected).
T.UNATTENDED_SESSION A user may gain unauthorized access to an unattended session.	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	O. ROBUST_TOE_ACCESS (FTA_SSL.1, FTA_SSL.2, FTA_SSL.3) helps to mitigate this threat by including mechanisms that place controls on user's sessions. Local administrator's sessions are locked and remote sessions are dropped after a Platform Administrator-defined time period of inactivity. Locking the local administrator's session reduces the opportunity of someone gaining unauthorized access the session when the console is unattended. Dropping the connection of a remote session (after the specified time period) reduces the risk of someone accessing the remote machine where the session was established, thus gaining unauthorized access to the session.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.UNAUTHORIZED_ACCESS A user may gain access to user data for which they are not authorized according to the TOE security policy.	O.MEDIATE The TOE must protect user data in accordance with its security policy.	O.MEDIATE (FDP_ACC.2, FDP_ACF.1) works to mitigate this threat by requiring that objects in the directory are protected using access control items. An access control item contains information about who is allowed to access an object, as well as the allowed modes of access. The settings present in the access control item selected in the access control decision process determine whether or not a user is authorized to access the object. It should be noted that multiple security policies can be (but do not *have* to be) in place in a single TOE, meaning that the process by which the target ACI is selected can be different for two different objects. It is required, however, that all objects be covered by this policy. Note that O.SELF_PROTECTION (FPT_RVM.1) ensures that this access control mechanism is always invoked, thus ensuring that users cannot bypass the mechanism to access data for which they are not authorized.
	O.USER_GUIDANCE The TOE will provide users with the information necessary to correctly use the security mechanisms.	O.USER_GUIDANCE (AGD_USR.1) mitigates this threat by providing the user the information necessary to use the security mechanisms that control access to user data in a secure manner. For instance, the method by which the discretionary access control mechanism (FDP_ACC.1, FDP_ACF.1) is configured, and how to apply it to the data the user owns, is described in the user guidance. If this information were not available to the user, the information may be left unprotected, or the user may mis-configure the controls and unintentionally allow unauthorized access to their data.

Threat/Policy	Objectives Addressing the Threat	Rationale
T.UNIDENTIFIED_ACTIONS The administrator may fail to notice potential security violations, thus limiting the administrator's ability to identify and take action against a possible security breach.	O.AUDIT_REVIEW The TOE will provide the capability to selectively view audit information, and alert the administrator of identified potential security violations.	O.AUDIT_REVIEW (FAU_SAA.1-NIAP-0407, FAU_ARP.1, FAU_ARP_ACK_DIR_(EXP).1, FAU_SAR.1, FAU_SAR.3) helps to mitigate this threat by providing a variety of mechanisms for monitoring the use of the system. The two basic ways audit review is performed is through analysis of the audit trail produced by the audit mechanism, and through the use of an automated analysis and alarm system. For analyzing the audit trail, the TOE requires an Auditor role. This role is restricted to Audit record review and the deletion of the audit trail for maintenance purposes. A search and sort capability provides an efficient mechanism for the Audit Administrator to view pertinent audit information. In addition to the local Auditor role, the TOE also has the capability to export the audit information to an external audit analysis tool (such as an intrusion detection system) for more detailed or composite audit analysis. The TOE's audit analysis mechanism must consist of a minimum set of configurable audit events that could indicate a potential security violation. Thresholds for these events must be configurable by an appropriate administrative role. By configuring these auditable events, the TOE monitors the occurrences of these events (e.g. set number of authentication failures, set number directory access failures, self-test failures, etc.) and immediately notifies an administrator once an event has occurred or a set threshold has been met.

Threat/Policy	Objectives Addressing the Threat	Rationale
		If a potential security violation has been detected, the TOE displays a message that identifies the potential security violation to all administrative consoles. The consoles include the local TOE console and any active remote directory administrator sessions. If an administrator is not currently logged into the TOE, the message is stored and immediately displayed the next time an administrator logs into the TOE. This message is displayed and will remain on the screen until an administrator acknowledges the message. At this point, all administrators that have received the message will receive notification that the alarm has been acknowledged, who acknowledged the alarm, and the time that it was acknowledged. In addition to displaying the potential security violation, the message must contain all audit records that generated the potential security violation. By enforcing the message content and display, this objective provides assurance that a TOE administrator will be notified of a potential security violation.
T.UNKNOWN_STATE When the TOE is initially started or restarted after a failure, the security state of the TOE may be unknown.	O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed.	O.MAINT_MODE (FPT_RCV.2) helps to mitigate this threat by ensuring that the TOE does not continue to operate in an insecure state when a hardware or software failure occurs. After a failure, the TOE enters a state that disallows operations and requires an administrator to follow documented procedures to return the TOE to a secure state.
	O.CORRECT_ TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	O.CORRECT_TSF_OPERATION (FPT_TST_(EXP).4, FPT_TST. _(EXP).5) counters this threat by ensuring that the TSF runs a suite of tests to successfully demonstrate the correct operation of the TSF (hardware and software) and the TSF's cryptographic components at initial startup of the TOE. In addition to ensuring that the TOE's security state can be verified, an administrator can verify the integrity of the TSF's data and stored code as well as the TSF's cryptographic data and stored code using the TOE-provided cryptographic mechanisms.

Threat/Policy	Objectives Addressing the Threat	Rationale
	O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.	O.SOUND_DESIGN (ADV_SPM.1) works to mitigate this threat by requiring that the TOE developers provide accurate and complete design documentation of the security mechanisms in the TOE, including a security model. By providing this documentation, the possible secure states of the TOE are described, thus enabling the administrator to return the TOE to one of these states during the recovery process.
	O. ROBUST_ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	O. ROBUST_ADMIN_GUIDANCE (ADO_IGS.1, AGD_ADM.1) provides administrative guidance for the secure start-up of the TOE as well as guidance to configure and administer the TOE securely. This guidance provides administrators with the information necessary to ensure that the TOE is started and initialized in a secure manor. The guidance also provides information about the corrective measure necessary when a failure occurs (i.e., how to bring the TOE back into a secure state).
P.ACCESS_BANNER The TOE shall display an initial banner describing restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE. Reference: DODI 8500.2 Enclosure 4, Attachment 4 ECWM-1 and ECAN-1	O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	O.DISPLAY_BANNER (FTA_TAB.1) satisfies this policy by ensuring that the TOE displays a Platform Administrator-configurable banner that provides all users with a warning about the unauthorized use of the TOE. This is required to be displayed before an interactive administrative session, since it does not make sense to display a banner for sessions involving directory requests from users, and those types of sessions are largely automated.

Threat/Policy	Objectives Addressing the Threat	Rationale
P.ACCOUNTABILITY The authorized users of the TOE shall be held accountable for their actions within the TOE. Reference: DODI 8500.2 Paragraph 5.12, Enclosure 4, Attachment 1,2,3, and 4, ECAT-2, ECRG-1, ECTP-1, ECAR-3, ECLC, etc.	O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security-relevant events associated with users.	O.AUDIT_GENERATION (FAU_GEN.1-NIAP-0407, FAU_GEN.2-NIAP-410, FIA_USB.1, FAU_SEL.1-NIAP-0407) addresses this policy by providing an audit mechanism to record the actions of a specific user, as well as the capability for an administrator to "pre-select" audit events based on the user ID. The audit event selection function is configurable during run-time to ensure the TOE is able to capture security-relevant events given changes in threat conditions. Additionally, the administrator's ID is recorded when any security relevant change is made to the TOE (e.g. access rule modification, start-stop of the audit mechanism, establishment of a trusted channel, etc.). Attributes used in the audit record generation process are also required to be bound to the subject, ensuring users are held accountable
	O.TIME_STAMPS The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.	O.TIME_STAMPS (FPT_STM.1, FMT_MTD.1) plays a role in supporting this policy by requiring the TOE to provide a reliable time stamp (configured locally by the Platform Administrator or via a trusted IT entity, such as an NTP server). The audit mechanism is required to include the current date and time in each audit record. All audit records that include the user ID will also include the date and time that the event occurred.
	O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	O. ROBUST_TOE_ACCESS (FIA_UID.2, FIA_UAU.2, FIA_UAU.5) supports this policy by requiring the TOE to identify and authenticate all authorized users prior to allowing any TOE access or any TOE mediated access on behalf of those users. Note that although the TSF allows access by anonymous users (FIA_UAU.1), this objective (and hence the policy) does not apply to such users because they are not authenticated.

Threat/Policy	Objectives Addressing the Threat	Rationale
P.ADMIN_ACCESS Administrators shall be able to administer the TOE both locally and remotely through protected communications channels. Reference: DODI 8500.2 Enclosure 4, Attachment 1,2, and 3 ECPA-1	O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	O.ADMIN_ROLE (FMT_SMR.2) supports this policy by requiring the TOE to provide mechanisms (e.g., local authentication, remote authentication, means to configure and manage the TOE both remotely and locally) that allow remote and local administration of the TOE. This is not to say that everything that can be done by a local administrator must also be provided to the remote administrator. In fact, it may be desirable to have some functionality restricted to the local administrator.
	O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication data.	O.TRUSTED_PATH (FTP_TRP.1, FTP_ITC.1) satisfies this policy by requiring that each remote administrative and management session for all trusted users is authenticated and conducted via a secure channel. Additionally, all trusted IT entities (e.g., trusted peer directories, intrusion detection systems) connect through a protected channel, thus avoiding disclosure and spoofing problems. This objective works in conjunction with the IT environment objective, OE.TRUSTED_PATH, each providing one end of the trusted channel.
P.CRYPTOGRAPHY The TOE shall use NIST FIPS validated cryptography as a baseline with additional NSA-approved methods for key management (i.e.; generation, access, distribution, destruction, handling, and storage of keys), and for cryptographic operations (i.e.; encryption, decryption, signature, hashing, key exchange, and random number generation services). Reference: DODI 8500.2 Enclosure 3, Paragraph E3.2.4.3.3	O.CRYPTOGRAPHY The TOE shall use NIST FIPS 140-2 validated cryptographic services.	O.CRYPTOGRAPHY_VALIDATED To be determined by the PP development team in collaboration with the cryptography support organization.
	O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	O.RESIDUAL_INFORMATION (FDP_RIP.2) counters this threat by ensuring that TSF data and user data is not persistent when resources are released by one user/process and allocated to another user/process. This means that network packets sent in response to a request will not have residual data from another packet (potentially from another user) due to the padding of a packet

Threat/Policy	Objectives Addressing the Threat	Rationale
P.VULNERABILITY_ ANALYSIS_TEST The TOE must undergo appropriate independent vulnerability analysis and penetration testing to demonstrate that the TOE is resistant to an attacker possessing a medium attack potential. Reference: DODI 8500.2 Enclosure 4, Attachment 5 ECMT-1	O.VULNERABILITY_ANALYSIS_ TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	O.VULNERABILITY_ANALYSIS_TEST (AVA_VLA.3) satisfies this policy by ensuring that an independent analysis is performed on the TOE and penetration testing based on that analysis is performed. Having an independent party perform the analysis helps ensure objectivity and eliminates preconceived notions of the TOE's design and implementation that may otherwise affect the thoroughness of the analysis. The level of analysis and testing requires that an attacker with a moderate attack potential cannot compromise the TOE's ability to enforce its security policies.

Appendix B: Mapping of Medium Robustness Objectives to Requirement

(Back to TOC)

Sample rationale is provided below. The PP authors should examine various NIAP evaluated PPs for examples of rationale.

Objectives	Requirements Addressing the Objective	Rationale
O.ADMIN_ROLE The TOE will provide administrator roles to isolate administrative actions, and to make the administrative functions available locally and remotely.	FMT_SMR	FMT_SMR.2 requires that three roles exist for administrative actions: the Security Administrator, who is responsible for configuring most security-relevant parameters on the TOE; the Cryptographic Administrator, who is responsible for managing the security data that is critical to the cryptographic operations; and the Auditor, who is responsible for reading and deleting the audit trail. The TSF is able to associate a human user with one or more roles and these roles isolate administrative functions in that the functions of these roles do not overlap. It is true that the design of some systems could enable a rogue security administrator to manipulate cryptographic data by, for instance, writing directly to kernel memory. While this scenario is a security concern, this objective does not counter that aspect of T.ADMIN_ROGUE. If a security administrator were to perform such an action, the auditing requirements (along with the audit trail protection requirements) afford some measure of detectability of the rogue platform administrator's actions.
O.AUDIT_GENERATION The TOE will provide the capability to detect and create records of security-relevant events associated with users.	FAU_GEN.1-NIAP-0407 FAU_GEN.2-NIAP-0410 FIA_USB.1 FAU_SEL.1-NIAP-0407	FAU_GEN.1-NIAP-0407 defines the set of events that the TOE must be capable of recording. This requirement ensures that an administrator has the ability to audit any security relevant event that takes place in the TOE. This requirement also defines the information that must be contained in the audit record for each auditable event. There is a minimum of information that must be present in every audit record and this requirement defines that, as well as the additional information that must be recorded for each auditable event. This requirement also places a requirement on the level of detail that is recorded on any additional security functional requirements an ST author

Objectives	Requirements Addressing the Objective	Rationale
		adds to this PP. FAU_GEN.2-NIAP-410 ensures that the audit records associate a user identity with the auditable event. Although the FIA_ATD.1 requirements mandate that a "userid " be used to represent a user identity, the TOE developer is able to associate different types of user-ids with different users in order to meet this objective. FAU_SEL.1-NIAP-0407 allows the selected administrator(s) to configure which auditable events will be recorded in the audit trail. This provides the administrator with the flexibility in recording only those events that are deemed necessary by site policy, thus reducing the amount of resources consumed by the audit mechanism and providing the ability to focus on the actions of an individual user. In addition, the requirement has been refined to require that the audit event selection function is configurable during run-time to ensure the TOE is able to capture security-relevant events given changes in threat conditions. FIA_USB.1 plays a role is satisfying this objective by requiring a binding of security attributes associated with users that are authenticated with the subjects that represent them in the TOE. This only applies to authenticated users, since the identity of unauthenticated users cannot be confirmed. Therefore, the audit trail may not always have the proper identity of the subject that causes an audit record to be generated (anonymous relying parties).
O.AUDIT_PROTECTION The TOE will provide the capability to protect audit information.	FMT_MOF FAU_SAR.2 FAU_STG.1-NIAP-0429 FAU_STG.3 FAU_STG-NIAP-0414-1	FMT_MOF.1 restricts the ability to control the behavior of the audit and alarm mechanism to the Security Administrator. The Security Administrator is the only user that controls the behavior of the events that generate alarms and whether the alarm mechanism is enabled or disabled. FAU_SAR.2 restricts the ability to read the audit trail to the Auditor, thus preventing the disclosure of the audit data to any other user. However, the TOE is not expected to prevent the disclosure of audit data if it has been archived or saved in another form (e.g., moved or copied to an ordinary file). The FAU_STG family dictates how the audit trail is protected. FAU_STG.1-NIAP-0429 restricts the ability to delete audit records to the

Objectives	Requirements Addressing the Objective	Rationale
		Auditor; or if the option of overwriting old audit records is chosen by the Platform/Directory Administrator in FAU_STG.NIAP-0414-1, the audit data may be deleted/overwritten. Since the auditor is trusted to review the audit data, the threat being countered is that the platform/directory administrator does something malicious and then attempts to conceal it by configuring the audit log to overwrite old records. Presumably the platform/directory administrator would then attempt to fill up the audit log in order to overwrite the thing they just did, as well as the fact that the they reconfigured the audit log overwrite action. The auditor would hopefully notice this activity and detect the fact that the platform/directory administrator was performing illicit activities. The fact that the platform/directory administrator does not directly have the ability to delete the audit records helps ensure that audit records are kept until the Auditor deems they are no longer necessary. FAU_STG.1-NIAP-0429 also ensures that no one has the ability to modify audit records (e.g., edit any of the information contained in an audit record). This ensures the integrity of the audit trail is maintained. FAU_STG.3 requires that the administrators be alerted when the audit trail exceeds a capacity threshold established by the Security Administrator. In addition, an audit record is cut which will trigger the analysis performed in FAU_SAA, resulting in an FAU_ARP alarm being issued. This ensures that an administrator has the opportunity to manage the audit trail before it becomes full and the avoiding the possible loss of audit data. FAU_STG.NIAP-0414-1 allows the Security Administrator to configure the TOE so that if the audit trail does become full, either the TOE will prevent any events from occurring (other than actions taken by the administrator) that would generate an audit record or the audit mechanism will overwrite the oldest audit records with new records. FMT_SMF.1 requires the TOE to provide an administrator with a facility to backup, recover and archive audit data ensuring the ability to recover corrupted audit records, and access to a complete history of audit information.
O.AUDIT_REVIEW The TOE will provide the capability to	FAU_ARP.1 FAU_ARP_ACK_(EXP).1	FAU_SAA.1-NIAP-0407 defines the events (or rules) that indicate a potential security violation and will generate an alarm. The triggers for

Objectives	Requirements Addressing the Objective	Rationale
selectively view audit information, and alert the administrator of identified potential security violations.	FAU_SAA.1-NIAP-0407 FAU_SAR.1 FAU_SAR.3	and will generate an alarm. The triggers for these events are largely configurable by the Security Administrator. Some rules are not configurable, or configurable by the cryptographic administrator. FAU_ARP.1 requires that the alarm be displayed at the local administrative console and at the remote administrative console(s) when auditor and security administrative session(s) exists. For alarms at remote consoles, the alarm is sent either during an established session or upon session establishment (as long as the alarm has not been acknowledged). This is required to increase the likelihood that the alarm will be received as soon as possible. This requirement also dictates the information that must be displayed with the alarm. The potential security violation is identified in the alarm, as are the contents of the audit records of the events that accumulated and triggered the alarm. The information in the audit records is necessary it allows the administrators to react to the potential security violation without having to search through the audit trail looking for the related events. FAU_ARP_ACK_(EXP).1 requires that an alarm generated by the mechanism that implements the FAU_ARP requirement be maintained until an administrator acknowledges it. This ensures that the alarm message will not be obstructed and the administrators will be alerted of a potential security violation. Additionally, this requires that the acknowledgement be transmitted to users that received the alarm, thus ensuring that that set of administrators knows that the user specified in the acknowledgement message has addressed the alarm. FAU_SAR.1 (both iterations) is used to provide both the auditor and an external audit analysis function the capability to read all the audit data contained in the audit trail. This requirement also mandates the audit information be presented in a manner that is suitable for the end user (auditor or external system) to interpret the audit trail. It is expected that the audit information be presented in such a way that the end user can examine an audit record and have the appropriate information (that required by FAU_GEN.2-NIAP-410) presented together to facilitate the analysis of the audit review.

Objectives	Requirements Addressing the Objective	Rationale
		Ensuring the audit data are presented in an interpretable format will enhance the ability of the entity performing the analysis to identify potential security violations. FAU_SAR.3 complements FAU_SAR.1 by providing the administrators the flexibility to specify criteria that can be used to search or sort the audit records residing in the audit trail. FAU_SAR.3 requires the administrators be able to establish the audit review criteria based on a userid and role so that the actions of a user can be readily identified and analyzed. Allowing the administrators to perform searches or sort the audit records based on dates and times provides the capability to facilitate the administrator's review of incidents that may have taken place at a certain time. It is important to note that the intent of sorting in this requirement is to allow the administrators the capability to organize or group the records associated with a given criteria.
O.CHANGE_MANAGEMENT The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled throughout the TOE's development.	ACM_AUT.1 ACM_CAP.4 ACM_SCP.2 ALC_DVS.1 ALC_FLR.2 ALC_LCD.1	ACM_CAP.4 contributes to this objective by requiring the developer have a configuration management plan that describes how changes to the TOE and its evaluation deliverables are managed. The developer is also required to employ a configuration management system that operates in accordance with the CM plan and provides the capability to control who on the development staff can make changes to the TOE and its developed evidence. This requirement also ensures that authorized changes to the TOE have been analyzed and the developer's acceptance plan describes how this analysis is performed and how decisions to incorporate the changes to the TOE are made ACM_SCP.2 is necessary to define what items must be under the control of the CM system. This requirement ensures that the TOE implementation representation, design documentation, test documentation (including the executable test suite), user and administrator guidance, CM documentation and security flaws are tracked by the CM system. ALC_DVS.1 requires the developer describe the security measures they employ to ensure the integrity and confidentiality of the TOE are maintained. The physical, procedural, and personnel security measures the developer uses provides an added level of control over who and how changes are made to the TOE and its

Objectives	Requirements Addressing the Objective	Rationale
		associated evidence. ALC_FLR.2 plays a role in satisfying the "analyzed" portion of this objective by requiring the developer to have procedures that address flaws that have been discovered in the product, either through developer actions (e.g., developer testing) or those discovered by others. The flaw remediation process used by the developer corrects any discovered flaws and performs an analysis to ensure new flaws are not created while fixing the discovered flaws. ALC_LCD.1 requires the developer to document the life-cycle model used in the development and maintenance of the TOE. This life-cycle model describes the procedural aspects regarding the development of the TOE, such as design methods, code or documentation reviews, how changes to the TOE are reviewed and accepted or rejected. ACM_AUT.1 complements ACM_CAP.4, by requiring that the CM system use an automated means to control changes made to the TOE. If automated tools are used by the developer to analyze, or track changes made to the TOE, those automated tools must be described. This aids in understanding how the CM system enforces the control over changes made to the TOE.
O.CORRECT_ TSF_OPERATION The TOE will provide a capability to test the TSF to ensure the correct operation of the TSF in its operational environment.	FPT_TST_(EXP).4, FPT_TST_(EXP).5	O_CORRECT_TSF_OPERATION requires two security functional requirements in the FPT class, FPT_TST. These functional requirements provide the end user with the capability to ensure the TOE's security mechanisms continue to operate correctly in the field. FPT_TST_(EXP).4 has been created to ensure end user tests exist to demonstrate the correct operation of the security mechanisms required by the TOE that are provided by the hardware and that the TOE's software and TSF data has not been corrupted. Hardware failures could render a TOE's software ineffective in enforcing its security policies and this requirement provides the end user the ability to discover any failures in the hardware security mechanisms. FPT_TST_(EXP).4 is necessary to ensure the correctness of the TSF software and TSF data. If TSF software is corrupted it is possible that the TSF would no longer be able to enforce the security policies. This also holds true for TSF data, if TSF data is corrupt the TOE may not correctly enforce its security policies.

Objectives	Requirements Addressing the Objective	Rationale
O.DISPLAY_BANNER The TOE will display an advisory warning regarding use of the TOE.	FTA_TAB.1	FTA_TAB.1 meets this objective by requiring the TOE display a Platform Administrator-defined banner before an administrator can establish an interactive session. This banner is under complete control of the Platform Administrator in which they specify any warnings regarding unauthorized use of the TOE and remove any product or version information if they desire.
O.DOCUMENT_KEY_LEAKAGE The bandwidth of channels that can be used to compromise key material shall be documented.	AVA_CCA_(EXP).2	AVA_CCA_(EXP).2 requires that a covert channel analysis be performed on the entire TOE to determine the bandwidth of possible cryptographic key leakage. While there are no requirements to limit the bandwidth, the results of this analysis will provide useful guidance on what the specified lifetime of the cryptographic keys should be in order to reduce the damage due to a key compromise.
O.MAINT_MODE The TOE shall provide a mode from which recovery or initial startup procedures can be performed.	FPT_RCV.2	This objective is met by using the FPT_RCV.2 requirement, which ensures that the TOE does not continue to operate in an insecure state when a hardware or software failure occurs. Upon the failure of the TSF self-tests the TOE will no longer be assured of enforcing its security policies. Therefore, the TOE enters a state that operations cease and requires an administrator to follow documented procedures that instruct them on to return the TOE to a secure state. These procedures may include running diagnostics of the hardware, or utilities that may correct any integrity problems found with the TSF data or code. Solely specifying that the administrator reload and install the TOE software from scratch, while might be required in some cases, does not meet the intent of this requirement.
O.MANAGE The TOE will provide all the functions and facilities necessary to support the administrators in their management of the security of the TOE, and restrict these functions and facilities from unauthorized use.	FMT_MTD FMT_MSA.1 FMT_MOF.1 FMT_SMF.1	The FMT requirements are used to satisfy this management objective, as well as other objectives that specify the control of functionality. The requirement's rationale for this objective focuses on the administrator's capability to perform management functions in order to control the behavior of security functions. FMT_MSA.1 provides the Security Administrator the capability to manipulate the security attributes of the objects in their scope of control that determine the access policy. There are several functions in the TSF that need to be enabled or disabled: either in a producer role or a consumer role; the ability to detect

Objectives	Requirements Addressing the Objective	Rationale
		attempts to replay operations sent by a relying party; and the ability to enable the cryptographic module self-tests to be run after generation of a key. The use of these functions is specified and restricted by the FMT_MOF.1 iterations. The following are examples of iterations of FMT_MTD.1 that were used by PP authors to satisfy some of the functions of O.MANAGE: The requirement FMT_MTD.1(1) is intended to be used by the ST author, with possible iterations, to address TSF data that has not already been specified by other FMT requirements. This is necessary because the ST author may add TSF data in assignments that cannot be addressed ahead of time by the PP authors. This requirement specifies that the manipulation of these data be restricted to the security administrator. FMT_MTD.1(2) provides the Cryptographic Administrator, and only the Cryptographic Administrator, the ability to modify the cryptographic security data. This allows the Cryptographic Administrator to change the critical data that affects the TOE's ability to perform its cryptographic functions properly. FMT_MTD.1(3) provides the capability of setting the date and time that is used to generate time stamps to the Security Administrator or a trusted IT entity (authorized data manager). It is important to allow this functionality, due to clock drift and other circumstances, but the capability must be restricted. A trusted IT entity is allowed in the selection made by the ST author to take in account the use of an NTP server or some other service that provides time information without human intervention. FMT_MTD.1(4) addresses the capabilities of data managers, who have responsibilities for security data management for sub-portions of the set of TSF data (for example, the platform clock time, sub-hierarchies of the directory). The scope of a data manager's responsibility is set by a security administrator, but they are expected to manage the entities in their scope of control without reliance on the security administrator. FMT_MTD.2(1), FMT_MTD.2(2) restrict the setting of limits on the processor time and network connection resources, respectively, to an administrator. This capability allows an

Objectives	Requirements Addressing the Objective	Rationale
		administrator to control the resources consumed by to provide a flexible policy with respect to denial of service attacks. The requirement FMT_SMF.1 was introduced as an international interpretation. This requirement specifies functionality that must be provided to administrators of the TOE. If the PP author includes this requirement care must be taken to use the other FMT requirements to specify how the functionality is restricted and to which role the functionality is provided.
O.MEDIATE The TOE must protect user data in accordance with its security policy.	FDP_ACC.2 FDP_ACF_1	The FDP_ACC.2 and FDP_ACF.1 requirements were chosen to define the policies, the subjects, objects, and operations for how and when mediation of access to the user data takes place. Because of the A.NO_GENERAL_PURPOSE assumption the no access control policy (for relying parties) needs to be defined for platform resources. FDP_ACC.2 specifies that the subjects under control of the policy are to be defined, and that all operations that involve access to (minimally) the data are controlled by the policy. These objects contain the user data to be protected. FDP_ACF.1 details the manner in which the user data are to be protected. The basics called for by the requirement is to match a set of attributes associated with a subject to a set of "access control items" associated with the object they wish to access; all applicable ACIs need to grant access in order for the subject to perform the operation on the object. The details of how the ACIs are collected and the specific operations supported are specified in the ST, and with the attributes define the security policy to be enforced. Setting the attributes (implementing the security policy) is a function of the administrator or system manager.
O.REPLAY_DETECTION The TOE will provide a means to detect and reject the replay of authentication data as well as other TSF data and security attributes.	FPT_RPL.1	The O.REPLAY_DETECTION objective is satisfied by FPT_RPL.1(1), which requires the TOE to detect and reject the attempted replay of authentication data from a remote user (administrator or relying party). This is sufficient to meet the objective because no untrusted users have local access to the TOE, thus there is no way to capture nor replay authentication data for a local session.
O.RESIDUAL_INFORMATION The TOE will ensure that any information contained in a protected resource is not released when the resource is reallocated.	FCS_CKM.2 FCS_CKM.4 FDP_RIP.2	FDP_RIP.2 is used to ensure the contents of resources are not available to subjects other than those explicitly granted access to the data. For this TOE it is critical that the memory used to build network packets containing replies to

Objectives	Requirements Addressing the Objective	Rationale
		relying party requests is either cleared or that some buffer management scheme be employed to prevent the contents of a packet being disclosed in a subsequent packet (e.g., if padding is used in the construction of a packet, it must not contain another user's data or TSF data).
O.RESOURCE_SHARING The TOE shall provide mechanisms that mitigate attempts to exhaust <*specific types of resources which the TOE protects*> resources provided by the TOE (e.g., *examples of resources that apply to technology*).	FRU_RSA.1 FMT_MTD.2 FMT_MOF.1	The following are examples of iterations of FMT_MTD.1 and FRU_RSA.1 that were used by PP authors to satisfy some of the functions of O.RESOURCE_SHARING: While an availability security policy does not explicitly exist, FRU_RSA.1 is used to mitigate potential resource exhaustion attempts. In order to mitigate the CPU exhaustion attempt, FRU_RSA.1(1) is included. This requires that the CPU time being consumed by a relying party must be limited to an amount specified by the security administrator (FMT_MTD.2(1)), and actions taken when an attempt is made are specified in FMT_MTD.2(1). This requirement takes into account all CPU resources being consumed by a user (relying party), and not just a single subject. FRU_RSA.1(2) was used to reduce the impact of an attempt being made to exhaust transport-layer representation implementation artifacts (e.g., the TCP "half-open connection" attack). This requirement indicates that a time period must exist when maximum quota (which is defined by the ST) is met or surpassed. Although this requirement (unlike the two previous requirements) does not mandate that the administrator be able to set this time period, FMT_MTD.2(2) restricts this functionality should the TOE implement it. FMT_MTD.2(2) also indicates (when filled in by the ST author) what action is to be taken when the quota is reached. FMT_MOF.1 dictates the functionality required to manage the security functions of the TOE. The ability to control this function is limited to the Security Administrator and provides this role the capability of enabling or disabling the function. This requirement also provides the Security Administrator with the capability to modify the behavior of the function that indicates a potential sharing violation. So as to ensure the mechanisms are configured as intended, the Security Administrator has the ability to view the conditions under which an sharing alarm will be generated, and if alarm

Objectives	Requirements Addressing the Objective	Rationale
		generation is enabled.
O. ADMIN_GUIDANCE The TOE will provide administrators with the necessary information for secure delivery and management.	ADO_DEL.2 ADO_IGS.1 AGD_ADM.1 AGD_USR.1 AVA_MSU.2	ADO_DEL.2 ensures that the administrator is provided documentation that instructs them how to ensure the delivery of the TOE, in whole or in parts, has not been tampered with or corrupted during delivery. This requirement ensures the administrator has the ability to begin their TOE installation with a *clean* (e.g., malicious code has not been inserted once it has left the developer's control) version of the TOE, which is necessary for secure management of the TOE. The ADO_IGS.1 requirement ensures the administrator has the information necessary to install the TOE in the evaluated configuration. Often times a vendor's product contains software that is not part of the TOE and has not been evaluated. The Installation, Generation and Startup (IGS) documentation ensures that once the administrator has followed the installation and configuration guidance the result is a TOE in a secure configuration. The AGD_ADM.1 requirement mandates the developer provide the administrator with guidance on how to operate the TOE in a secure manner. This includes describing the interfaces the administrator uses in managing the TOE, security parameters that are configurable by the administrator, how to configure the TOE's rule set and the implications of any dependencies of individual rules. The documentation also provides a description of how to setup and review the auditing features of the TOE. The AGD_USR.1 is intended for non-administrative users, but could be used to provide guidance on security that is common to both administrators and non-administrators (e.g., password management guidelines). Since the non-administrative users of this TOE are limited to relying parties it is expected that the user guidance would discuss how the data validation authentication mechanism is used, and any instructions on authenticating to the TOE. The description of the use of these mechanisms would not have to be repeated in the administrator's guide. AVA_MSU.2 ensures that the guidance documentation is complete and can be followed unambiguously to ensure the TOE is not mis-configured in an insecure state due to confusing guidance.

Objectives	Requirements Addressing the Objective	Rationale
O.ROBUST_TOE_ACCESS The TOE will provide mechanisms that control a user's logical access to the TOE and to explicitly deny access to specific users when appropriate.	FIA_UID.2 FIA_AFL.1 FIA_ATD.1 FIA_UAU.1 FIA_UAU.2 FIA_UAU.5 FTA_TSE.1 AVA_SOF FTA_SSL.1 FTA_SSL.2 FTA_SSL.3 AVA_SOF.1	The following are examples of iterations of FIA_UAU.1 that were used by PP authors to satisfy some of the functions of O.ROBUST_TOE_ACCESS: FIA_UID.2 plays a small role in satisfying this objective by ensuring that every user is identified before the TOE performs any mediated functions. FIA_ATD.1 defines the attributes of users, including a userid that is used to by the TOE to determine a user's identity and enforce what type of access the user has to the TOE (e.g., the TOE associates a userid with any role(s) they may assume). This requirement allows a human user to have more than one user identity assigned, so that a single human user could assume all the roles necessary to manage the TOE. In order to ensure a separation of roles, this PP requires a single role to be associated with a user id. This is inconvenient in that the administrator would be required to log in with a different user id each time they wish to assume a different role, but this helps mitigate the risk that could occur if an administrator were to execute malicious code. FIA_UAU.1(1) contributes to this objective by limiting the services that are provided by the TOE to unauthenticated users. Management requirements and the unauthenticated information flow policy requirement provide additional control on these services. FIA_UAU.1(2) identifies the services that are provided by the TOE that do not require authentication. The inclusion of this requirement does not restrict who has logical access to the TOE, and therefore poses additional risk exposure. FIA_UAU.2 requires that administrators authorized IT entities and other users authenticate themselves to the TOE before performing administrative duties (including those performed by authorized IT entities (e.g., NTP server)), or using the services identified in this requirement.. In order to control logical access to the TOE an authentication mechanism is required. The explicit requirement FIA_UAU_(EXP).5 mandates that the TOE provide a local authentication mechanism. This requirement

Objectives	Requirements Addressing the Objective	Rationale
		also affords the ST author the opportunity to add additional authentication mechanisms (e.g., single-use, certificates) if they desire. Local authentication is required to ensure someone that has physical access to the TOE and has not been granted logical access (e.g., a janitor) cannot gain unauthorized logical access to the TOE. The AVA_SOF.1 requirement is applied to the local authentication mechanism. For this TOE, the strength of function specified is medium. This requirement ensures the developer has performed an analysis of the authentication mechanism to ensure the probability of guessing a user's authentication data would require a high-attack potential, as defined in Annex B of the CEM. FTA_TSE.1.1 contributes to this objective by limiting a user's ability to logically access the TOE. This requirement provides the Security Administrator the ability to control when (e.g., time and day(s) of the week) and where (e.g., from a specific network address) remote administrators can access the TOE. FIA_AFL.1 provides a detection mechanism for unsuccessful authentication attempts by remote administrators, authenticated proxy users and authorized IT entities. The requirement enables a Security Administrator settable threshold that prevents unauthorized users from gaining access to authorized user's account by guessing authentication data by locking the targeted account until the Security Administrator takes some action (e.g., re-enables the account) or for some Security Administrator defined time period. Thus, limiting an unauthorized user's ability to gain unauthorized access to the TOE. The FTA_SSL family partially satisfies the O. TOE_ACCESS objective by ensuring that user's sessions are afforded some level of protection. FTA_SSL.1 provides the Security Administrator the capability to specify a time interval of inactivity in which an unattended local administrative session would be locked and will require the administrator responsible for that session to re-authenticate before the session can be used to access TOE resources. FTA_SSL.2 provides administrators the ability to lock their local administrative session. This component allows administrators to protect their session immediately, rather than waiting for the time-out

Objectives	Requirements Addressing the Objective	Rationale
		period and minimizes their session's risk of exposure. FTA_SSL.3 takes into account remote sessions. After a Security Administrator defined time interval of inactivity remote sessions will be terminated, this includes user proxy sessions and remote administrative sessions. This component is especially necessary, since remote sessions are not typically afforded the same physical protections that local sessions are provided.
O.SELF_PROTECTION The TSF will maintain a domain for its own execution that protects itself and its resources from external interference, tampering or unauthorized disclosure.	FPT_SEP.2 FPT_RVM.1	FPT_SEP was chosen to ensure the TSF provides a domain that protects itself from untrusted users. If the TSF cannot protect itself it cannot be relied upon to enforce its security policies. FPT_SEP.1 could have been used to address the previous notion, however, FPT_SEP.2 was used to require that the *cryptographic module* be provided its own address space. This is necessary to reduce the impact of programming errors in the remaining portions of the TSF on the cryptographic module. The inclusion of FPT_RVM.1 ensures that the TSF makes policy decisions on all interfaces that perform operations on subjects and objects that are scoped by the policies. Without this non-bypassability requirement, the TSF could not be relied upon to completely enforce the security policies, since an interface(s) may otherwise exist that would provide a user with access to TOE resources (including TSF data and executable code) regardless of the defined policies. This includes controlling the accessibility to interfaces, as well as what access control is provided within the interfaces.
O.SOUND_DESIGN The TOE will be designed using sound design principles and techniques. The TOE design, design principles and design techniques will be adequately and accurately documented.	ADV_FSP_(EXP).1 ADV_HLD_(EXP).1 ADV_INT_(EXP).1 ADV_LLD_(EXP).1 ADV_ARC_(EXP).1 ADV_RCR.1 ADV_SPM.1	There are two different perspectives for this objective. One is from the developer's point of view and the other is from the evaluator's. The ADV class of requirements is levied to aide in the understanding of the design for both parties, which ultimately helps to ensure the design is sound. ADV_INT_(EXP).1 ensures that the design of the TOE has been performed using good software engineering design principles that require a modular design of the TSF. Modular code increases the developer's understanding of the interactions within the TSF, which in turn, potentially reduces the amount of errors in the design. Having a modular design is imperative for evaluator's to gain an appropriate level of

Objectives	Requirements Addressing the Objective	Rationale
		understanding of the TOE's design in a relatively short amount of time. The appropriate level of understanding is dictated by other assurance requirements in this PP (e.g., ATE_DPT.2, AVA_CCA_(EXP).2, AVA_VLA.3). ADV_SPM.1 requires the developer to provide an informal model of the security policies of the TOE. Modeling these policies helps understand and reduce the unintended side effects that occur during the TOE's operation that might adversely affect the TOE's ability to enforce its security policies. ADV_FSP_(EXP).1 requires that the interfaces to the TSF be completely specified. In this TOE, a complete specification of the network interface (including the network interface card) is critical in understanding what functionality is presented to untrusted users and how that functionality fits into the enforcement of security policies. Some network protocols have inherent flaws and users have the ability to provide the TOE with network packets crafted to take advantage of these flaws. The routines/functions that process the fields in the network protocols allowed (e.g., TCP, UPD, ICMP, directory-specific protocols such as LDAP) must fully specified: the acceptable parameters, the errors that can be generated, and what, if any, exceptions exist in the processing. The functional specification of the hardware interface (e.g., network interface card) is also extremely critical. Any processing that is externally visible performed by NIC must be specified in the functional specification. Having a complete understanding of what is available at the TSF interface allows one to analyze this functionality in the context of design flaws. ADV_HLD_(EXP).1 requires that a high-level design of the TOE be provided. This level of design describes the architecture of the TOE in terms of subsystems. It identifies which subsystems are responsible for making and enforcing security relevant (e.g., anything relating to an SFR) decisions and provides a description, at a high level, of how those decisions are made and enforced. Having this level of description helps provide a general understanding of how the TOE works, without getting buried in details, and may allow the reader to discover flaws in the design. ADV_ARC_(EXP).1 addresses the non-

Objectives	Requirements Addressing the Objective	Rationale
		bypassability (FPT_RVM) and domain separation (FPT_SEP) aspects of the TSF, since these need to be analyzed differently from other functional requirements. The low-level design, as required by ADV_LLD_(EXP).1, provides the reader with the details of the TOE's design and describes at a module level how the design of the TOE addresses the SFRs. This level of description provides the detail of how modules interact within the TOE and if a flaw exists in the TOE's design, it is more likely to be found here rather than the high-level design. This requirement also mandates that the interfaces presented by modules be specified. Having knowledge of the parameters a module accepts, the errors that can be returned and a description of how the module works to support the security policies allows the design to be understood at its lowest level. ADV_RCR.1 is used to ensure that the levels of decomposition of the TOE's design are consistent with one another. This is important, since design decisions that are analyzed and made at one level (e.g., functional specification) that are not correctly designed at a lower level may lead to a design flaw. This requirement helps in the design analysis to ensure design decisions are realized at all levels of the design.
O.SOUND_IMPLEMENTATION The implementation of the TOE will be an accurate instantiation of its design, and is adequately and accurately documented.	ADV_IMP.2 ADV_INT_(EXP).1 ADV_LLD_(EXP).1 ADV_ARC_(EXP).1 ADV_RCR.1 ALC_TAT.1	While ADV_LLD_(EXP).1 (and ADV_ARC_(EXP).1 for the FPT_SEP and FPT_RVM aspects of the TSF) is used to aide in ensuring that the TOE's design is sound, it also contributes to ensuring the implementation is correctly realized from the design. It is expected that evaluators will use the low-level design as an aide in understanding the implementation representation. The low-level design requirements ensure the evaluators have enough information to intelligently analyze (e.g., the documented interface descriptions of the modules match the entry points in the module, error codes returned by the functions in the module are consistent with those identified in the documentation) the implementation and ensure it is consistent with the design. While evaluators have the ability to "negotiate" the subset in ADV_IMP.1, ADV_IMP.2 was chosen to ensure evaluators have full access to the source code. If the evaluators are limited in their ability to analyze source code they may not be able to determine the accuracy of the implementation or the adequacy of the

Objectives	Requirements Addressing the Objective	Rationale
		documentation. Often times it is difficult for an evaluator to identify the complete sample of code they wish to analyze. Often times looking at code in one subsystem may lead the evaluator to discover code they should look at in another subsystem. Rather than require the evaluator to "re-negotiate" another sample of code, the complete implementation representation is required. When performing the activities associated with the ADV_INT_(EXP).1 requirement, the evaluators will ensure that the architecture of the implementation is modular and consistent with the architecture presented in the low-level design. Having a modular implementation provides the evaluators with the ability to more easily assess the accuracy of the implementation, with respect to the design. If the implementation is overly complex (e.g., circular dependencies, not well understood coupling, reliance on side-effects) the evaluator may not have the ability to assess the accuracy of the implementation. ALC_TAT.1 provides evaluators with information necessary to understand the implementation representation and what the resulting implementation will consist of. Critical areas (e.g., the use of libraries, what definitions are used, compiler options) are documented so the evaluator can determine how the implementation representation is to be analyzed. ADV_RCR.1 is used here to provide the correspondence of the lowest level of decomposition (e.g., source code) to the adjoining level, low-level design. The correspondence analysis is used by the evaluator as a tool when determining if the low-level design is correctly reflected in the implementation representation
O.THOROUGH_FUNCTIONAL_TESTING The TOE will undergo appropriate security functional testing that demonstrates the TSF satisfies the security functional requirements.	ATE_COV.2 ATE_FUN.1 ATE_IND.2 ATE_DPT.2	In order to satisfy O.THOROUGH_FUNCTIONAL_TESTING, the ATE class of requirements is necessary. The component ATE_FUN.1 requires the developer to provide the necessary test documentation to allow for an independent analysis of the developer's security functional test coverage. In addition, the developer must provide the test suite executables and source code, which are used for independently verifying the test suite results and in support of the test coverage analysis activities. ATE_COV.2 requires the developer to provide a test coverage analysis

Objectives	Requirements Addressing the Objective	Rationale
		that demonstrates the TSFI are completely addressed by the developer's test suite. While exhaustive testing of the TSFI is not required, this component ensures that the security functionality of each TSFI is addressed. This component also requires an independent confirmation of the completeness of the test suite, which aids in ensuring that correct security relevant functionality of a TSFI is demonstrated through the testing effort. ATE_DPT.2 requires the developer to provide a test coverage analysis that demonstrates depth of coverage of the test suite. This component complements ATE_COV.2 by ensuring that the developer takes into account the high-level and low-level design when developing their test suite. Since exhaustive testing of the TSFI is not required, ATE_DPT.2 ensures that subtleties in TSF behavior that are not readily apparent in the functional specification are addressed in the test suite. ATE_IND.2 requires an independent confirmation of the developer's test results, by mandating a subset of the test suite be run by an independent party. This component also requires an independent party to attempt to craft functional tests that address functional behavior that is not demonstrated in the developer's test suite. Upon successful adherence to these requirements, the TOE's conformance to the specified security functional requirements will have been demonstrated.
O.TIME_STAMPS The TOE shall provide reliable time stamps and the capability for the administrator to set the time used for these time stamps.	FPT_STM.1 FMT_MTD.1	FPT_STM.1 requires that the TOE be able to provide reliable time stamps for its own use and therefore, partially satisfies this objective. Time stamps include date and time and are reliable in that they are always available to the TOE, and the clock must be monotonically increasing. The following is an example of an iteration of FMT_MTD.1 that was used by a PP author to satisfy the function of O.TIME_STAMPS. FMT_MTD.1(3) satisfies the rest of this objective by providing the capability to set the time used for generating time stamps to either the Security Administrator, trusted IT entity, or both. The authorized IT entity was included as an option for the possible use of an NTP server to set the TOE's time.
O.TRUSTED_PATH The TOE will provide a means to ensure that users are not communicating with some other entity pretending to be the TOE when supplying identification and authentication	FTP_TRP FTP_ITC	FTP_TRP.1.1 requires the TOE to provide a mechanism that creates a distinct communication path that protects the data that traverses this path from disclosure (first iteration) or modification (second iteration).

Objectives	Requirements Addressing the Objective	Rationale
supplying identification and authentication data.		This requirement ensures that the TOE can identify the end points and ensures that a user cannot insert themselves between the user and the TOE, by requiring that the means used for invoking the communication path cannot be intercepted and allow a "man-in-the-middle-attack" (this does not prevent someone from capturing the traffic and replaying it at a later time – see FPT_RPL.1). Since the user invokes the trusted path (FTP_TRP.1.2) mechanism they can be assured they are communicating with the TOE. FTP_TRP.1.3 mandates that the trusted path be the only means available for providing identification and authentication information, therefore ensuring a user's authentication data will not be compromised when performing authentication functions. Furthermore, the remote administrator's communication path is encrypted during the entire session. The following are examples of iterations of FTP_ITC.1 and FTP_ITC.1 that were used by PP authors to satisfy some of the functions of O.TRUSTED_PATH. FTP_ITC.1(1) and FTP_ITC.1(2) are similar to FTP_TRP.1(1) and FTP_TRP.1(2), in that they require a mechanism that creates a distinct communication path with the same characteristics, however FTP_ITC.1(1) and FTP_ITC.1(2) is used to protect communications between IT entities, rather than between a human user and an IT entity. FTP_ITC.1.3 requires the TOE to initiate the trusted channel, which ensures that the TOE has established a communication path with an authorized IT entity and not some other entity pretending to be an authorized IT entity.
O.USER_GUIDANCE The TOE will provide users with the information necessary to correctly use the security mechanisms.	AGD_USR.1	The user guidance required by AGD_USR.1 meets the objective by describing the discretionary access controls available to the user, and how to set the attributes pertaining to the mechanism. This guidance also instructs the user how to log on to the TOE, and how to choose passwords that will not be easily compromised through a brute force attack.
O.VULNERABILITY_ANALYSIS_TEST The TOE will undergo appropriate independent vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with medium attack potential to violate the TOE's security policies.	AVA_VLA.3	To maintain consistency with the overall assurance goals of this TOE, O.VULNERABILITY_ANALYSIS_TEST requires the AVA_VLA.3 component to provide the necessary level of confidence that vulnerabilities do not exist in the TOE that could cause the security policies to be violated. AVA_VLA.3 requires the developer to perform

Objectives	Requirements Addressing the Objective	Rationale
		a systematic search for potential vulnerabilities in all the TOE deliverables. For those vulnerabilities that are not eliminated, a rationale must be provided that describes why these vulnerabilities cannot be exploited by a threat agent with a moderate attack potential, which is in keeping with the desired assurance level of this TOE. As with the functional testing, a key element in this component is that an independent assessment of the completeness of the developer's analysis is made, and more importantly, an independent vulnerability analysis coupled with testing of the TOE is performed. This component provides the confidence that security flaws do not exist in the TOE that could be exploited by a threat agent of moderate (or lower) attack potential to violate the TOE's security policies.

Appendix C: Sample PP Mapping Spreadsheet

(Back to TOC)

As mentioned in the main body of the this guidance, it is helpful to keep track of the mapping between the threats/policies in the PP, the objectives that contribute to the mitigation of each threat and implementation of each policy, and the specific requirements from each objective that apply to each threat or component. While the PPRB recommends that the PP authors make a working copy of Table 7 and update it while they are working on the PP, Table 7 takes up many pages and it is sometimes difficult to get an overall view of the mappings. The PPRB has found that a spreadsheet provides this condensed view and proved useful in writing consistent PP according to the medium robustness CIM. As noted in the main text of this guidance, the spreadsheet is nothing more than Table 7 without the notes column or all of the text associated with each threat and objective. Additionally, it is not expected that the spreadsheet be part of the PP; it is instead a tool for the PP authors to use or not, as they wish. An example spreadsheet that is associated with this CIM is provided below.

Threats/Policies	Objectives	CC Function and Security Requirements					
T.ADMIN_ERROR	O.ROBUST_ADMIN_GUIDEANCE	ADO_DEL.2	ADO_IGS.1	AGD_ADM.1	AGD_USR.1	AVA_MSU.2	
	O.ADMIN_ROLE	FMT_SMR					
	O.MANAGE	FMT_MTD					
T.ADMIN_ROGUE	O.ADMIN_ROLE	FMT_SMR					
T.AUDIT_COMPROMISE	O.AUDIT_PROTECTION	FMT_MOF	FAU_SAR.2	FAU_STG.1-NIAP-0429	FAU_STG.3	FAU_STG.NIAP-0414-1	
	O.RESIDUAL_INFORMATION	FDP_RIP.2					
	O.SELF_PROTECTION	FPT_SEP.2	FPT_RVM.1				
T.CRYPTO_COMPROMISE	O.RESIDUAL_INFORMATION	To Be determined by the PP developers					
	O.SELF_PROTECTION						
	O.DOCUMENT_KEY_LEAKAGE						
T.EAVESDROP	O.PROTECT_IN_TRANSIT	FDP_ITT.1	FPT_ITT.1				
T.MASQUERADE	O.ROBUST_TOE_ACCESS	FIA_AFL.1	FIA_ATD.1	FIA_UID	FIA_UAU	FTA_TSE.1	AVA_SOF
T.FLAWED_IMPLEMENTATION	O.CHANGE_MANAGEMENT	ACM_AUT.1	ACM_CAP.4	ACM_SCP.2	ALC_DVS.1	ALC_FLR.2	ALC_LCD.1
	O.THOROUGH_FUNCTIONAL_TESTING	ARE_COV.2	ATE_FUN.1	ATE_DPT.2	ATE_IND.2		
	O.SOUND_IMPLEMENT	ADV_FSP.2	ADV_HLD.2	ADV_INT.1	ADV_LLD.1	ADV_RCR.1	ADV_SPM.

Threats/Policies	Objectives	CC Function and Security Requirements					
	ATION						1
	O.VULNERABILITY_ANALYSIS_TEST	AVA_VLA.3					
T.POOR_TEST	O.CORRECT_TSF_OPERATION	FPT_AMT.1	FPT_TST				
	O.THOROUGH_FUNCTIONAL_TESTING	ATE_COV.2	ATE_FUN.1	ATE_IND.2	ATE_DPT.2		
	O.VULNERABILITY_ANALYSIS_TEST	AVA_VLA.3					
T.REPLAY	O.REPLAY_DETECTION	FPT_RPL.1					
T.RESIDUAL_DATA	O.RESIDUAL_INFORMATION	FDP_RIP.2	FCS_CKM.2	FCS_CKM.4			
T.RESOURCE_EXHAUSTION	O.RESOURCE_SHARING	FRU_RSA.1	FMT_MTD.2	FMT_MOF.1			
T.SPOOFING	O.TRUSTED_PATH	FTP_TRP	FTP_ITC				
T.MALICIOUS_TSF_COMPROMISE	O.RESIDUAL_INFORMATION	FDP_RIP.2	FCS_CKM.2	FCS_CKM.4			
	O.SELF_PROTECTION	FPT_SEP.2	FPT_RVM.1				
	O.MANAGE	FMT_MTD.1	FMT_MSA.1	FMT_MOF.1	FMT_SMF.1		
	O.DISPLAY_BANNER	FTA_TAB.1					
	O.TRUSTED_PATH	FTP_TRP	FTP_ITC				
T.UNATTENDED_SESSION	O.ROBUST_TOE_ACCESS	FTA_SSL.1	FTA_SSL.2	FTA_SSL.3	AVA_SOF.1		
T.UNAUTHORIZED_ACCESS	O.MEDIATE	FDP_ACC	FDP_ACF	FDP.IFF			
T.UNIDENTIFIED_ACTIONS	O.AUDIT_REVIEW	FAU_ARP.1	FAU_ARP_ACK_(EXP).1	FAU_SAA.1-NIAP-0407	FAU_SAR.1	FAU_SAR.3	
T.UNKNOWN_STATE	O.MAINT_MODE	FPT_RCV.2					
	O.CORRECT_TSF_OERATION	FPT_AMT.1	FPT_TST				
	O.SOUND_DESIGN	ADV_SPM.1					
	O.ROBUST_ADMIN_GUIDEANCE	ADO_IGS.1	AGD_ADM.1				
P.ACCESS_BANNER	O.DISPLAY_BANNER	FTA_TAB.1					
P.ACCOUNTABILITY	O.AUDIT_GENERATION	FAU_GEN.1-NIAP-0407	FAU_GEN.2-NIAP-410	FIA_USB.1	FAU_SEL.1-NIAP-0407		
	O.TIME_STAMPS	FPT_STM.1	FMT_MTD.1				
	O.ROBUST_TOE_ACCESS	FIA_UID					
P.ADMIN_ACCESS	O.ADMIN_ROLE	FMT_SMR					

Threats/Policies	Objectives	CC Function and Security Requirements					
	O.TRUSTED_PATH	FTP_TRP	FTP_ITC				
P.CRYPTOGRAPHY	O.CRYPTOGRAPHY	To be determined by the PP developer	In collaboration with cryptographic support organization				
	O.RESIDUAL_INFORMATION						
P.VULNERABILITY_ANALYSIS_TEST	O.VULNERABILITY_ANALYSIS_TEST	AVA_VLA.3					

Appendix D: Explanatory Material for Explicit Assurance Requirements

ADV_ARC_(EXP).1 Architectural design
ADV_INT_(EXP).1 Modular decomposition
ADV_FSP_(EXP).1 Functional specification With Complete Summary,
ADV_HLD_(EXP).1 Security-enforcing high-level design
ADV_LLD_(EXP).1 Security-enforcing low-level design

PP Appendix for ADV_INT_(EXP)

(Back to TOC)

This explicit component was created to levy different modularity metrics on the SFP-enforcing modules and non-SFP-enforcing modules.

The parts of the TSF that implement an SFP (in this component, SFP-enforcing is used to designate modules that enforce an SFP) that is determined and assigned by the PP/ST author, are those modules that interact (defined in the coupling analysis) with the module or modules that provide the TSFI for that SFP with justified exceptions. The intent is that all of the modules that play an SFR related role (as opposed to modules that provide infrastructure support, such as scheduling, reading binary data from the disk) in enforcing an SFP are identified as SFP-enforcing. The remaining modules in the TSF are deemed non-SFP-enforcing modules, since they could be TSP-enforcing (e.g., enforcing a policy not assigned to this component), as well as TSP-supporting.

Objectives

This component addresses the internal structure of the software TSF. The SFP-enforcing modules require stricter adherence to the coupling and cohesion metrics than the metrics levied on the non-SFP-enforcing modules due to their key role in policy enforcement. While the non-SFP-enforcing modules also play a role in enforcing policy, their role is not as critical as the SFP-enforcing modules, therefore, the degree of coupling and cohesion required of these modules is not as restrictive. It is expected that all of the TSF modules are designed using good software engineering practice, whether they are developed by the developer or incorporated as a third party implementation into the TSF.

Requirements are presented for modular decomposition of the SFP-enforcing and non-SFP-enforcing functionality within the TSF. These requirements, when applied to the internal structure of the TSF, should result in improvements that aid both the developer and the evaluator in understanding the TSF, and also provides the basis for designing and evaluating test suites. Further, improving understandability of the TSF should assist the developer in simplifying its maintainability. The principal goal achieved by inclusion of the requirements from the ADV_INT class in a PP/ST is understandability of the TSF.

Modular design aids in achieving understandability by clarifying what dependencies and interactions a module has on other modules (*coupling*), by including in a module only tasks that are strongly related to each other (*cohesion*), and by illuminating the design of a module by using internal structuring and reduced complexity. The use of modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Its use enhances clarity of design and provides for increased assurance that unexpected effects do not occur. Additional desirable properties of modular decomposition are a reduction in the amount of redundant or unneeded code.

The incorporation of modular decomposition into the design and implementation process must be accompanied by sound software engineering considerations. A practical, useful software system will usually entail some undesirable coupling among modules, some modules that include loosely-related functions, and some subtlety or complexity in a module's design. These deviations from the ideals of modular decomposition are often deemed necessary to achieve some goal or constraint, be it related to performance, compatibility, future planned functionality, or some other factors, and may be acceptable, based on the developer's justification for them. In applying the requirements of this class, due consideration must be given to sound software engineering principles; however, the overall objective of achieving understandability must be achieved.

Another key component to reducing complexity is the use of coding standards. Coding standards are used as a reference to ensure programmers generate code that can be easily understood by individuals (e.g., code maintainers, code reviewers, evaluators) that are not intimately familiar with the nuances of the functions performed by the code. For example, coding standards ensure that meaningful names are given to variables and data structures, the code has a structure that is similar to code developed by other programmers, loops used in the code are understandable (e.g., leaving a loop to another section of code and returning is undesirable), the use of pointers to variables/data structures is straightforward, and the code is suitably commented (inline and/or by a preamble). The use of coding standards helps to eliminate errors in code development and maintenance, and assists the development team in performing code walk-throughs. Some aspects of coding standards are specific to a given program language (e.g., the C language may have a different standard than the Java language or assembly level code). It is expected that the coding standards are appropriately followed for the employed programming language(s). The requirements in this component allow for exceptions to the adherence of coding standards that may be necessary for reasons of performance, or some other factors, but these deviations must be justified (on a per module basis) as to why they are necessary. Any justification provided must address why the deviation does not unduly introduce complexity into the module, since ultimately, the goal of adhering to coding standards is to improve clarity.

Design complexity minimization is a key characteristic of a reference validation mechanism, the purpose of which is to arrive at a TSF that is easily understood so that it can be completely analyzed. (There are other important characteristics of a reference validation mechanism, such as TSF self-protection and TSP non-bypassability; these other characteristics are covered by requirements from other classes.)

Application notes

Several of the elements within this component refer to the architectural description. The architectural description is at a similar level of abstraction as the low-level design, in that it is concerned with the modules of the TSF. Whereas the low-level design describes the design of the modules of the TSF, the purpose of the architectural description is to provide evidence of modular decomposition of the TSF. Both the low-level design and the implementation representation are required to be in compliance with the architectural description, to provide assurance that these TSF representations possess the required modular decomposition.

This component requires the PP or ST author to fill in an assignment with the SFPs that are felt to be critical to the TOE and therefore their resulting design and implementation require stricter metrics for modularity. The SFPs can be those explicitly identified in the CC (i.e., FDP_ACC, FDP_IFF) by simply placing the appropriate label as specified in those requirements, or other policies determined by the PP/ST author (e.g., I&A, Audit), in which case, the PP/ST author should explicitly identify all of the SFRs that they intend to satisfy a policy that is not explicitly stated in the CC. This is necessary since currently a convention does not exist to place a convenient label on these policies.

The requirements in this component refer to SFP-enforcing and non-SFP-enforcing portions of the TSF. The non-SFP-enforcing portions of the TSF consist of the TSP-supporting modules and TSP-enforcing modules that do not play a role in the enforcement of the SFP(s) identified in ADV_INT_(EXP).1.4D as depicted in the Figure D1, where is this example, non-SFP-enforcing is everything in the TSF other than the SFP-enforcing functions.

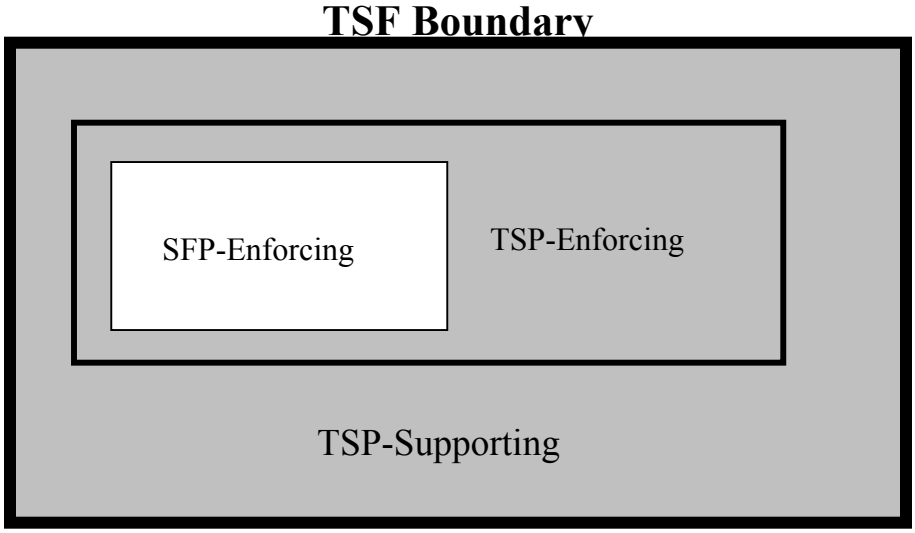


Figure D1. SFP-enforcing may only be a subset of TSP-enforcing functions.

The developer is required to identify the modules that are SFP-enforcing and implicitly the remaining modules, which will be non-SFP-enforcing. As stated earlier, the SFP-enforcing modules are those modules that interact with the module or modules that provide the TSFI for that SFP with justified exceptions. The justification of the non-SFP-enforcing modules (ADV_INT_(EXP).1.3C) is required only for those modules that interact with SFP-enforcing modules and not for all non-SFP-enforcing modules. As

depicted in the Figure D2 below, if a TSFI has already been designated as non-SFP-enforcing then the designation of the modules interacting with the module providing the TSFI do not have to be justified (e.g., modules X, Y, Z). The justification of the designation is only necessary for the module(s) that interact with a module that provides a TSFI that is SFP-enforcing (e.g., modules D, E, F (since it is writing to a global variable that Module A is reading, but in this example, it is not an SFP-enforcing variable).

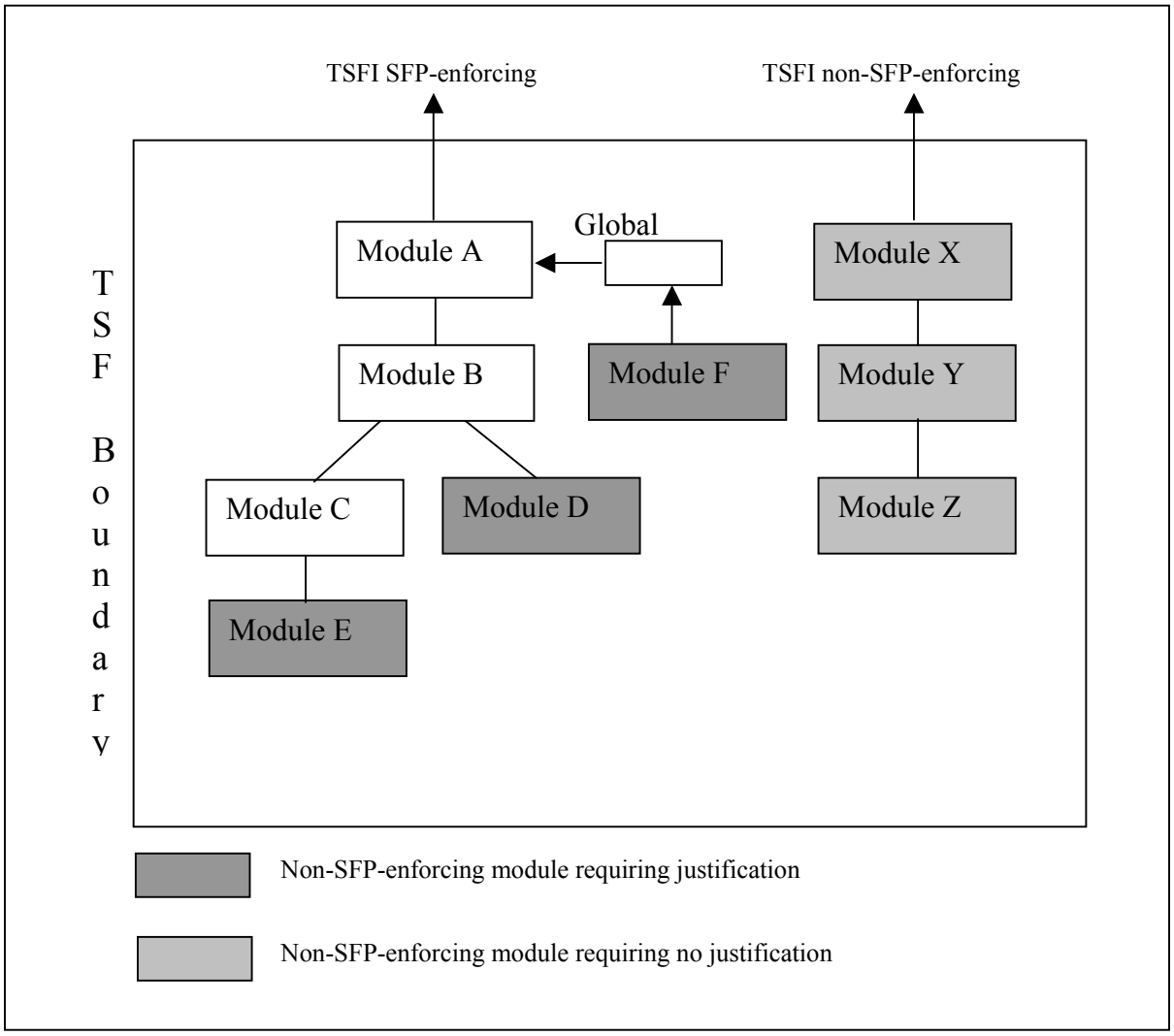


Figure D2. Example of non-SFP-enforcing modules requiring justification.

The modules identified in the architectural description are the same as the modules identified in the low-level design.

Terms, definitions and background

The following terms are used in the requirements for software internal structuring. Some of these are derived from the Institute of Electrical and Electronics

Engineers *Glossary of software engineering terminology, IEEE Std 610.12-1990.*

module: one or more source code files that cannot be decomposed into smaller compliable units.

modular decomposition: the process of breaking a system into components to facilitate design and development.

*cohesion (*also called *module strength):* the manner and degree to which the tasks performed by a single software module are related to one another; types of cohesion include coincidental, communicational, functional, logical, sequential, and temporal. These types of cohesion are characterized below, listed in the order of decreasing desirability.

functional cohesion: a module with this characteristic performs activities related to a single purpose. A functionally cohesive module transforms a single type of input into a single type of output, such as a *stack manager* or a *queue manager*.

sequential cohesion: a module with this characteristic contains functions each of whose output is input for the following function in the module. An example of a sequentially cohesive module is one that contains the functions to write audit records and to maintain a running count of the accumulated number of audit violations of a specified type.

communicational cohesion: a module with this characteristic contains functions that produce output for, or use output from, other functions within the module. An example of a communicationally cohesive module is an *access check* module that includes mandatory, discretionary, and capability checks.

temporal cohesion: a module with this characteristic contains functions that need to be executed at about the same time. Examples of temporally cohesive modules include *initialization, recovery,* and *shutdown* modules.

logical (or procedural) cohesion: a module with this characteristic performs similar activities on different data structures. A module exhibits logical cohesion if its functions perform related, but different, operations on different inputs.

coincidental cohesion: a module with this characteristic performs unrelated, or loosely related activities.

coupling: the manner and degree of interdependence between software modules; types of coupling include call, common and content coupling. These types of coupling are characterized below, listed in the order of decreasing desirability

call: two modules are call coupled if they communicate strictly through the use of their documented function calls; examples of call coupling are data, stamp, and control, which are defined below.

- *data:* two modules are data coupled if they communicate

strictly through the use of call parameters that represent single data items.

- *stamp:* two modules are stamp coupled if they communicate through the use of call parameters that comprise multiple fields or that have meaningful internal structures.
- *control:* two modules are control coupled if one passes information that is intended to influence the internal logic of the other.

common: two modules are common coupled if they share a common data area or a common system resource. Global variables indicate that modules using those global variables are common coupled.[13]

Common coupling through global variables is generally allowed, but only to a limited degree. For example, variables that are placed into a global area, but are used by only a single module, are inappropriately placed, and should be removed. Other factors that need to be considered in assessing the suitability of global variables are:

The number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification must be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them.

The number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.

content: two modules are content coupled if one can make direct reference to the internals of the other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that some or all of the content of one module are effectively included in the other. Content coupling can be thought of as using unadvertised module interfaces; this is in contrast to call coupling, which uses only advertised module interfaces.

---

[13] *It can be argued that modules sharing definitions, such as data structure definitions, are common coupled. However, for the purposes of this analysis, shared definitions are considered acceptable, but are subject to the cohesion analysis.*

call tree: a diagram that identifies the modules in a system and shows which modules call one another. All the modules named in a call tree that originates with (i.e., is rooted by) a specific module are the modules that directly or indirectly implement the functions of the originating module.

software engineering: the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. As with engineering practices in general, some amount of judgment must be used in applying engineering principles. Many factors affect choices, not just the application of measures of modular decomposition, layering, and minimization. For example, a developer may design a system with future applications in mind that will not be implemented initially. The developer may choose to include some logic to handle these future applications without fully implementing them; further, the developer may include some calls to as-yet unimplemented modules, leaving *call stubs*. The developer's justification for such deviations from well-structured programs will have to be assessed using judgment, as well as the application of good software engineering discipline.

complexity: this is a measure of how difficult software is to understand, and thus to analyze, test, and maintain. Reducing complexity is the ultimate goal for using modular decomposition, layering and minimization. Controlling coupling and cohesion contributes significantly to this goal.

A good deal of effort in the software engineering field has been expended in attempting to develop metrics to measure the complexity of source code. Most of these metrics use easily computed properties of the source code, such as the number of operators and operands, the complexity of the control flow graph (*cyclomatic complexity*), the number of lines of source code, the ratio of comments to executable code, and similar measures. Coding standards have been found to be a useful tool in generating code that is more readily understood.

While this component calls for the evaluator to perform a *complexity analysis*, it is expected that the developer will provide support for the claims that the modules are not overly complex (ADV_INT_(EXP).1.3D, ADV_INT_(EXP).1.6D, ADV_INT_(EXP).1.9C). This support could include the developer's programming standards, and an indication that all modules meet the standard (or that there are some exceptions that are justified by software engineering arguments). It could include the results of tools used to measure some of the properties of the source code. Or it could include other support that the developer finds appropriate.

PP Appendix for ADV_FSP_(EXP).1

(Back to TOC)

The functional specification is a description of the user-visible interface to the TSF. It contains an instantiation of the TOE security functional requirements. The functional specification has to completely address all of the user-visible TOE security functional requirements.

Application Notes

A description of the TSF interfaces (TSFI) provides fundamental evidence on which assurance in the TOE can be built. Fundamentally, the functional specification provides a description of *what* the TSF provides to users (as opposed to the high-level design and low-level design, which provide a description of *how* the functionality is provided). Further, the functional specification provides this information in the form of interface (TSFI) documentation.

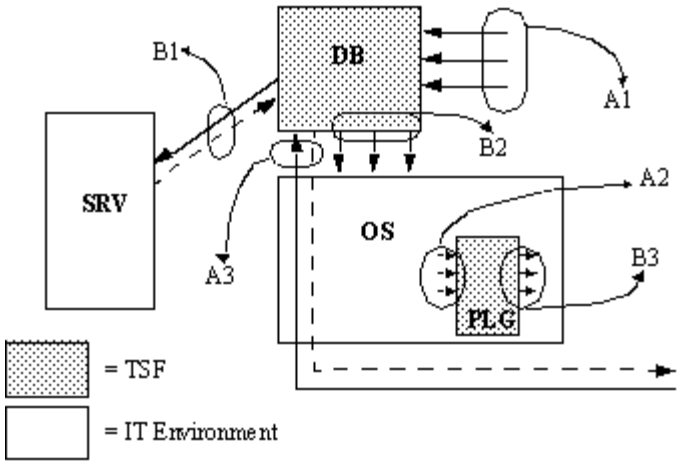
In order to identify the software interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of ADV_HLD_(EXP) analysis. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

1. The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST. This is typically all software that runs in a privileged state of the underlying hardware, as well as software that runs in unprivileged states that performs security functionality.

2. The software used by administrators in order to perform security management activities specified in the guidance documentation. These activities are a superset of those specified by any FMT_* functional requirements in the ST.

Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI are interfaces *to* the security services/resources the TSF is providing. This is especially relevant for TSFs that have dependencies on the IT environment, because not only is the TSF providing security services (and thus exposing TSFI), but it is also *using* services of the IT environment. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence to integrators and consumers of the system, and thus documentation requirements for these interfaces are specified in ADV_INT.

This concept (and concepts to be discussed in the following paragraphs) is illustrated in the following figure.



The figure above illustrates a TOE (a database management system) that has dependencies on the IT environment. The shaded boxes represent the TSF, while the un-shaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labeled "DB") and a kernel module that runs as part of the OS that performs some security function (represented by the box labeled "PLG"). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labeled "OS") itself, as well as an external server (labeled SRV). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labeled Ax for TSFI, and Bx for interfaces to be documented in ADV_INT. Each of these groups of interfaces is now discussed.

Interface group A1 represents the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

Interface group A3 represents TSFI that "pass through" the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

Non-TSFI interfaces pictured are labeled Bx. Interface group B1 is the most complex of these, because the architecture of the system and environmental assumptions and conditions will drive its analysis. In the first case, assume that, either through an environmental assumption or an IT environmental requirement, the network link between the DB and SRV is protected (it could be on a separate subnet, or it could be protected by a firewall such that only the DB could connect to the port on the SRV) such that only the DB has access to the SRV. In this case, the interface needs only to be documented in the integrator guidance, since untrusted users are unable to gain access.

However, consider the case where SRV is now just "somewhere on the network", and now the port that the DB opens up to communicate with the SRV is "exposed" to untrusted users. In this case, while the interface presented by the DB (the TSF) still only needs to be documented in the integrator guidance, additional considerations with respect to vulnerabilities may need to be documented as part of the AVA_VLA activity because of this exposure.

In the course of performing its functions, the DB will make system calls down to the OS.

This is represented by interface group B2. While these calls are not part of the TSFI, they are an interface that needs to be documented in the integrator guidance.

Interface group B3, mentioned previously in connection with interface group A2, is similar to interface group B2 in that these are calls made by the TSF to the IT environment to perform services for the TSF.

Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion categorizes the TSFI into the two categories mentioned previously: TSFI to software directly implementing the SFRs, and TSFI used by administrators.

TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the *.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an "additional" requirement that the functions that an administrator uses to perform their duties—as documented in administrative guidance—also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

> The administrative tool used is also accessible to untrusted users, and runs with some "privilege" itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.
>
> The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (AGD_ADM, including FMT_* actions) are part of the TSFI. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view. Also note that when FPT_SEP is included in the ST, the executable image of such tools need to be protected so that an untrusted user cannot replace the tool with a "trojan" tool.
>
> The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool when FPT_SEP is included in the ST.

It is also important to note that some TOEs will have interfaces that one might consider

part of the TSFI, but environmental factors remove them from consideration (an example is the case of interface group B1 discussed earlier). Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration, no firewall-provided services such as telnet). Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the system, or they could use a GUI-based tool that essentially translated the GUI-based checkboxes, textboxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

The term “administrator” above is used in the sense of an entity that has complete trust with respect to all policies implemented by the TSF. There may be entities that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an “administrator”, they need to be treated as untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a “wrapper” interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered “interfaces.” Switches that can change the behavior of the hardware are also part of the interface.

As indicated above, an interface exists at the TSF boundary if it can be used (by an administrator; untrusted user; or another TOE) to affect the behavior of the TSF. The requirements in this family apply to all types of TSFI, not just APIs.

All TSFI are *security relevant*, but some interfaces (or aspects of interfaces) are more critical and require more analysis than other interfaces. If an interface plays a role in enforcing any security policy on the system, then that interface is *security enforcing*. Such policies are not limited to the access control policies, but also refer to any functionality provided by one of the SFRs contained in the ST (with exceptions for FPT_SEP and FPT_RVM as detailed below). Note that it is possible that an interface may have various effects and exceptions, some of which may be security enforcing and some of which may not.

FPT_SEP and FPT_RVM are SFRs that require a different type of analysis from other SFRs. These requirements are architecturally related, and their implementation (or lack thereof) is not easily (or efficiently) testable at the TSFI. From a terminology standpoint, although implementation (and the associated analysis) of FPT_SEP and FPT_RVM is

critical to the trustworthiness of the system, these two SFRs will not be considered as SFRs that are applicable when determining the set of security-enforcing TSFIs as defined in the previous paragraph.

Interfaces (or parts of an interface) that need only to function correctly in order for the security policies of the system to be preserved are termed *security supporting*. A security supporting interface typically plays a role in supporting the architectural requirements (FPT_SEP or FPT_RVM), meaning that as long as it can be shown that it does not allow the TSF to be compromised or bypassed no further analysis against SFRs is required. In order for an interface to be security supporting it must have *no* security enforcing aspects. In contrast, a security enforcing interface may have security supporting aspects (for example, the ability to set the system clock may be a security enforcing aspect of an interface, but if that same interface is used to display the system date that effect may only be security supporting).

A key aspect for the assurance associated with this component is the concept of the evaluator being able to verify that the developer has correctly categorized the security enforcing and security supporting interfaces. The requirements are structured such that the information required for security supporting interfaces is the *minimum* necessary in order for the evaluator to make this determination in an effective manner.

For the purposes of the requirements, interfaces are specified (in varying degrees of detail) in terms of their parameters, parameter descriptions, effects, exceptions, and error messages. Additionally, the purpose of each interface, and the way in which the interface is used (both from the point of view of the external stimulus (e.g., the programmer calling the API, the administrator changing a setting in the registry) and the effect on the TSFI that stimulus has) must be specified. This description of method of use must also specify how those administrative interfaces that are unable to be successfully invoked by untrusted users (case “c” mentioned above) are protected.

Parameters are explicit inputs to and outputs from an interface that control the behavior of that interface. For examples, parameters are the arguments supplied to an API; the various fields in a packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

A parameter description tells what the parameter is in some meaningful way. For instance, the interface “foo(i)” could be described as having “parameter i which is an integer”; this is not an acceptable parameter description. A description such as “parameter i is an integer that indicates the number of users currently logged in to the system.” is required.

Effects of an interface describe what the interface does. The effects that need to be described in an FSP are those that are visible at any external interface, not necessarily limited to the one being specified. For instance, the sole effect of an API call is not just the error code it returns. Also, depending on the parameters of an interface, there may be many different effects (for instance, an API might have the first parameter be a “subcommand”, and the following parameters be specific to that subcommand. The IOCTL API in some Unix systems is an example of such an interface).

Exceptions refer to the processing associated with "special checks" that may be performed by an interface. An example would be an interface that has a certain set of effects for all users except the Superuser; this would be an exception to the normal effect of the interface. Use of a privilege for some kind of special effect would also be covered in this topic.

Documenting the errors associated with the TSF is not as straightforward as it might appear, and deserves some discussion. A general principle is that errors generated by the TSF that are visible to the user should be documented. These errors can be the direct result of invoking a TSFI (an API call that returns an error); an indirect error that is easily tied to a TSFI (setting a parameter in a configuration that is error-checked when read, returning an immediate notification); or an indirect error that is not easily tied to a TSFI (setting a parameter that, in combination with certain system states, generates an error condition that occurs at a later time. An example might be resource exhaustion of a TSF resource due to setting a parameter to too low of a value).

Errors can take many forms, depending on the interface being described. For an API, the interface itself may return an error code; set a global error condition, or set a certain parameter with an error code. For a configuration file, an incorrectly configured parameter may cause an error message to be written to a log file. For a hardware PCI card, an error condition may raise a signal on the bus, or trigger an exception condition to the CPU.

For the purposes of the requirements, errors are divided into two categories. The first category includes *direct errors,* which are directly related to a TSFI; examples are API calls and parameter-checking for configuration files. For this category of errors, the functional specification must document all of the errors that can be returned as a result of invoking a security-enforcing aspect of the interface such that a reader should be able to associate an interface with the errors it is capable of generating. The second category includes *indirect errors*, which are errors that are not directly tied to the invocation of a TSFI, but which are reported to the user as a result of processing that occurs in the TSF. It should be noted that while the condition that causes the indirect error can be documented; it is generally much harder to document all the ways in which that condition can occur.[14] Because of the difficulty associated with documenting all of the ways to cause an error, and because of the cost of documenting all indirect errors compared to the benefit of having them documented, indirect errors are not required to be documented.

The ADV_FSP_(EXP).1.2E element defines a requirement that the evaluator determines that the functional specification is an accurate and complete instantiation of the TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the functional specification, in addition to the pairwise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the functional specification.

---

[14]*This may even be impossible, if the error message is for a condition that the programmer does not expect to occur, but is inserted as part of "defensive programming."*

PP Appendix for ADV_HLD_(EXP).1

(Back to TOC)

The high-level design of a TOE provides both context for a description of the TSF, and a thorough description of the TSF in terms of major structural units (i.e. subsystems). It relates these units to the functions that they provide. The high-level design requirements are intended to provide assurance that the TOE provides an architecture appropriate to implement the security-enforcing TOE security functional requirements.

To provide context for the description of the TSF, the high-level design describes the entire TOE at a high level. From this description the reader should be able to distinguish between the subsystems that are part of the TSF and those that are not. The remainder of the high-level design document then describes the TSF in more detail.

The high-level design refines the functional specification into subsystem descriptions. The functional specification provides a description of *what* the TSF does at its interface; the high-level design provides more insight into the TSF by describing *how* the TSF works in order to perform the functions specified at the TSFI. For each subsystem of the TSF, the high-level design identifies the TSFI implemented in the subsystem, describes the purpose of the subsystem and how the implementation of the TSFI (or portions of the TSFI) is designed. The interrelationships of subsystems are also defined in the high-level design. These interrelationships will be represented as data flows, control flows, etc. among the subsystems. It should be noted that this description is at a high level; low-level implementation detail is not necessary at this level of abstraction.

The developer is expected to describe the design of the TSF in terms of subsystems. The term “subsystem” is used here to express the idea of decomposing the TSF into a relatively small number of parts. While the developer is not required to actually have “subsystems”, the developer is expected to represent a similar level of decomposition. For example, a design may be similarly decomposed using “layers”, “domains”, or “servers”.

A security enforcing subsystem is a subsystem that provides mechanisms for enforcing an element of the TSP, or directly supports a subsystem that is responsible for enforcing the TSP. If a subsystem provides a security-enforcing interface, then the subsystem is security enforcing. If a subsystem does not provide any security enforcing TSFIs, its mechanisms still must preserve the security of the TSF; such subsystems are termed security supporting.

As was the case with ADV_FSP_(EXP), the set of SFRs that determine the TSP for the purposes of this component do not include FPT_SEP and FPT_RVM. Those two architectural functional requirements require a different type of analysis than that needed for all other SFRs. A security-enforcing subsystem is one that is designed to implement an SFR other than FPT_SEP and FPT_RVM; the design information and justification for the FPT_SEP and FPT_RVM requirements is given as a result of the ADV_ARC_(EXP) component.

The ADV_HLD_(EXP) component requires that the developer must identify all subsystems of the TSF (not just the security-enforcing ones). In general, the component requires that the security-enforcing aspects of the subsystems be described in more detail than the security-supporting aspects. The descriptions for the security-enforcing aspects should provide the reader with enough information to determine *how* the implementation of the SFRs is designed, while the description for the security-supporting aspects should provide the reader enough assurance to determine that 1) all security-enforcing behavior has been identified and 2) the subsystems or portions of subsystems that are security supporting have been correctly classified.

The ADV_HLD_(EXP).1.2E element for this component defines a requirement that the evaluator determine that the high-level design is an accurate and complete instantiation of the user-visible TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the high-level design, in addition to the pair wise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the high-level design. Note that for this element FPT_SEP and FPT_RVM are not explicitly analyzed; the analysis for those requirements is done as part of the activity for the ADV_ARC_(EXP) component.

PP Appendix for ADV_LLD_(EXP).1

(Back to TOC)

The low-level design of a TOE provides a description of the internal workings of the TSF in terms of modules, global data, and their interrelationships. The low-level design is a description of *how* the TSF is implemented to perform its functions, rather than *what* the TSF provides as is specified in the FSP. The low-level design is closely tied to the actual implementation of the TSF, unlike the high-level design, which could be implementation-independent. The primary goal of the low-level design is an aid in understanding the implementation of the TSF, both by reviewing the text of the low-level design as well as a guide when examining the implementation representation (source code).

A module is generally a relatively small architectural unit that exhibits properties discussed in ADV_INT_(EXP). A “module” in terms in of the ADV_LLD_(EXP) requirement refers to the same entity as a “module” for the ADV_INT_(EXP) requirement.

A security-enforcing module is a module that directly implements a security-enforcing TSFI. While this could, for example, include all modules in the call-tree of a security-enforcing module, typically there will be some modules in the call-tree of a security-enforcing module that are not themselves security enforcing. If a module of the TSF is not security enforcing, its implementation still must preserve the security of the TSF; such modules are termed security supporting.

A description of a security-enforcing module in the low-level design should be of sufficient detail so that one could create an implementation of the module from the low-level design, and that implementation would

1. be identical to the actual TSF implementation in terms of the interfaces presented and used by the module, and

2. be algorithmically identical to the implementation of the module. For instance, the low-level design may describe a block of processing that is looped over a number of times. The actual implementation may be a *for* loop or a *do* loop, both of which could be used to implement the algorithm. Likewise, a collection of objects could be represented by a linked list or an array; this level of detail is not required to be presented, since both are algorithmically identical. Conversely, if a module's actual implementation performed a bubble sort, it would be inadequate for the low-level design to specify that the module "performed a sort"; it would have to describe the type of sort that was being performed.

Security-supporting modules do not need to be described in the same amount of detail, but they should be identified and enough information should be supplied so that 1) the evaluation team can determine that such modules are correctly classified as security supporting (vs. security enforcing), and 2) the evaluation team has the information necessary to complete the analysis required by ADV_INT_(EXP).1.

In the low-level design, security-enforcing modules are described in terms of the interfaces they present to other modules; the interfaces they use (call interfaces) from other modules; global data they access; their purpose; and an algorithmic description of how they provide that function. Security supporting modules are described only in terms of the interfaces they present and their purpose.

The interfaces presented by a module are those interfaces used by other modules to invoke the functionality provided. Interfaces are described in terms of how their parameters, and any values that are returned from the interface. In addition to a list of parameters, the descriptions of these parameters are also given. If a parameter were expected to take on a set of values (e.g., a "flag" parameter), the complete set of values the parameter could take on that would have an effect on module processing would be specified. Likewise, parameters representing data structures are described such that each field of the data structure is identified and described. Note that different programming languages may have additional "interfaces" that would be non-obvious; an example would be operator/function overloading in C++. This "implicit interface" in the class description would also be described as part of the low-level design. Note that although a module could present only one interface, it is more common that a module presents a small set of related interfaces.

By contrast, interfaces used by a module must be identified such that it can be determined the unique interface that is being invoked by the module being described. It must also be clear from the low-level design the algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be "Module A invokes the double_bubble() interface in Module B to perform a bubble sort." An adequate

algorithmic description would be "Module A invokes the double_bubble routine with the list of access control entries; double_bubble() will return the entries sorted first on the username, then on the access_allowed field according the following rules..." The low-level design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting these called interfaces is via a call tree, and then the algorithmic description can be included in the algorithmic description of the called module.

If the implementation makes use of global data, the low-level design must describe the global data, and in the algorithmic descriptions of the modules indicate how the specific global data are used by the module. Global data are identified and described much like parameters of an interface.

The purpose a module fulfills is a short description indicating what function the module provides. The level of detail provided should be such that the reader could get a general idea of what the module's function is in the architecture, and to determine (for security-supporting modules) that it is not a security-enforcing module.

As discussed previously, the algorithmic description of the module should describe in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or informal text. It discusses how the parameters to the interface, global data, and called functions are used to accomplish the result. It notes changes to global data, system state, and return values produced by the module. It is at the level of detail that an implementation could be derived that would be very similar to the actual implementation of the system. It does not need to describe actual implementation artifacts (*do* loops vs *for* loops, linked lists vs arrays) if such artifacts are algorithmically identical.

It should be noted that source code does not meet the low-level design requirements. Although the low-level design describes the implementation, it *is not* the implementation. Further, the comments surrounding the source code are not sufficient low-level design if delivered interspersed in the source code. The low-level design must stand on its own, and not depend on source code to provide details that must be provided in the low level design (whether intentionally or unintentionally). However, if the comments were extracted by some automated or manual process to produce the low-level design (independent of the source code statements), they could be found to be acceptable if they met all of the appropriate requirements.

The ADV_LLD_(EXP).1.2E element in this component defines a requirement that the evaluator determine that the low-level design is an accurate and complete instantiation of the user-visible TOE security functional requirements. This provides a direct correspondence between the TOE security functional requirements and the low-level design, in addition to the pair-wise correspondences required by the ADV_RCR family. Although the evaluator may use the evidence provided in ADV_RCR as an input to making this determination, ADV_RCR cannot be the basis for a positive finding in this area. The requirement for completeness is intended to be relative to the level of abstraction of the low-level design. Note that for this element, FPT_SEP and FPT_RVM are not explicitly analyzed; the analysis for those requirements is done as part of the activity for the ADV_ARC_(EXP) component.

PP Appendix for ADV_ARC_(EXP).1

(Back to TOC)

The architectural design of the TOE is related to the information contained in other decomposition documentation (functional specification, high-level design, low-level design) provided for the TSF, but presents the design in a manner that supports the argument that the TSP cannot be compromised (FPT_SEP) and that it cannot be bypassed (FPT_RVM). The objective of this component is for the developer to provide an architectural design and justification associated with the integrity and non-bypassability properties of the TSF.

FPT_SEP and FPT_RVM are distinct from other SFRs because they largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are achieved through the design of the system, and enforced by the correct implementation of that design. Because of their pervasive nature, the material needed to provide the assurance that these requirements are being achieved is better suited to a presentation separate from the design decomposition of the TSF as embodied in ADV_FSP_(EXP), ADV_HLD_(EXP), and ADV_LLD_(EXP). This is not to imply that the architectural design called for by this component cannot reference or make use of the design composition material; but it is likely that much of the detail present in the decomposition documentation will not be relevant to the argument being provided for the architectural design document.

The architectural design document consists of two types of information. The first is the design information for the entire TSF related to the FPT_SEP and FPT_RVM requirements. This type of information, like the decompositions for ADV_HLD_(EXP) and ADV_FSP_(EXP), describes *how* the TSF is implemented. The description, however, should be focused on providing information sufficient for the reader to determine that the TSF implementation is likely not to be compromised, and that the TSP enforcement mechanisms (that is, those that are implementing SFRs other than FPT_SEP and FPT_RVM) are likely always being invoked.

The nature of the FPT_SEP requirement lends itself to a design description much better than FPT_RVM. For FPT_SEP, mechanisms can be identified (e.g., memory management, protected processing modes provided by the hardware, etc.) and described that implement the domain separation. However, FPT_RVM is concerned with interfaces that bypass the enforcement mechanisms. In most cases this is a consequence of the implementation, where if a programmer is writing an interface that accesses or manipulates an object, it is that programmer's responsibility to use interfaces that are part of the TSP enforcement mechanism for the object and not to try to "go around" those interfaces. However, the developer is still able to describe architectural elements (e.g., object managers, macros to be invoked for specific functionality) that pertain to the design of the system to achieve the "always invoked" property of the TSF.

For FPT_SEP, the design description should cover how user input is handled by

privileged-mode routine; what hardware self-protection mechanisms are used and how they work (e.g., memory management hardware, including translation lookaside buffers); how software portions of the TSF use the hardware self-protection mechanisms in providing their functions; and any software protection constructs or coding conventions that contribute to meeting FPT_SEP.

For FPT_RVM, the description should cover resources that are protected under the SFRs (usually FDP_* components) and functionality (e.g., audit) that is provided by the TSF. The description should also identify the interfaces that are associated with each of the resources or the functionality; this might make use of the information in the FSP. This description should also describe any design constructs, such as object managers, and their method of use. For instance, if routines are to use a standard macro to produce an audit record, this convention is a part of the design that contributes to the non-bypassability of the audit mechanism. It's important to note that "non-bypassability" in this context is not an attempt to answer the question "could a part of the TSF implementation, if malicious, bypass a TSP mechanism", but rather it's to document how the actual implementation does not bypass the mechanisms implementing the TSP.

In addition to the descriptive information indicated in the previous paragraphs, the second type of information an architectural design document must contain is a justification that the FPT_SEP and FPT_RVM requirements are being met. This is distinct from the description, and presents an argument for why the design presented in the description is sufficient.

For FPT_SEP, the justification should cover the possible modes by which the TSF could be compromised, and how the mechanisms implemented in response to FPT_SEP counter such compromises. The vulnerability analysis might be referenced in this section.

For FPT_RVM, the justification demonstrates that whenever a resource protected by an SFR is accessed, the protection mechanisms of the TSF are invoked (that is, there are no "backdoor" methods of accessing resources that are not identified and analyzed as part of the ADV_FSP_(EXP)/ADV_HLD_(EXP)/ADV_LLD_(EXP) analysis). Similarly, the description demonstrates that a function described by an SFR is always provided where required. For example, if the FCO_NRO family were being used the description should demonstrate that all interfaces either 1) do not deal with transmitting the information identified in the FCO_NRO component included in the ST, or 2) invoke the mechanism(s) described by the decomposition documentation. The justification for FPT_RVM will likely need to address all of the TSFI in order to make the case that the TSP is non-bypassable.

Appendix E: PP Cover Sheet Template

(Back to TOC)

An example cover sheet is provided below and should be used as a template by the author of the PP. The author shall replace the [Technology Area] with the technology area of the PP. In addition, the date and version number of the profile should also be included.

US Government Protection Profile:

[Technology Area]

For

Medium Robustness Environments

Information Assurance Directorate

Month dd, yyyy
Version x.x

Appendix F: CC Abbreviations and Glossary

Back to the TOC

CC Abbreviations

The following abbreviations are common to more than one part of the CC:

CC Common Criteria
EAL Evaluation Assurance Level
IT Information Technology
PP Protection Profile
SF Security Function
SFP Security Function Policy
SOF Strength of Function
ST Security Target
TOE Target of Evaluation
TSC TSF Scope of Control
TSF TOE Security Functions
TSFI TSF Interface
TSP TOE Security Policy

Back to the TOC

CC Glossary (see the CC Part 1 for a complete list)

Assignment — The specification of an identified parameter in a component.
Assurance — Grounds for confidence that an entity meets its security objectives.
Class — A grouping of families that share a common focus.
Component — The smallest selectable set of elements that may be included in a PP, an ST, or a package.
Dependency — A relationship between requirements such that the requirement that is depended upon must normally be satisfied for the other requirements to be able to meet their objectives.
Developer – Those individuals (e.g., engineers, integrators, ISSOs, ISSEs) that generate evidence for evaluation in accordance with the CC requirements.
Element — An indivisible security requirement.
Evaluation — Assessment of a PP, an ST or a TOE, against defined criteria.
Evaluation Assurance Level (EAL) — A package consisting of assurance components from Part 3 that represents a point on the CC predefined assurance scale.
Evaluator -- An expert who will examine and judge the evidence carefully and in accordance with the CC.
Extension — The addition to an ST or PP of functional requirements not contained in Part 2 and/or assurance requirements not contained in Part 3 of the CC.
Family — A grouping of components that share security objectives but may differ in emphasis or rigor.
Iteration — The use of a component more than once with varying operations.
Object — An entity within the TSC that contains or receives information and upon which subjects perform operations.
Protection Profile (PP) — An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs.

Refinement — The addition of details to a component.
Role — A predefined set of rules establishing the allowed interactions between a user and the TOE.
Secret — Information that must be known only to authorized users and/or the TSF in order to enforce a specific SFP.
Security attribute — Information associated with subjects, users and/or objects that is used for the enforcement of the TSP.
Security Function (SF) — A part or parts of the TOE that have to be relied upon for enforcing a closely related subset of the rules from the TSP.
Security Function Policy (SFP) — The security policy enforced by an SF.
Security Target (ST) — A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Selection — The specification of one or more items from a list in a component.
System — A specific IT installation, with a particular purpose and operational environment.
Target of Evaluation (TOE) — An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
TOE resource — Anything useable or consumable in the TOE.
TOE Security Functions (TSF) — A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.
TOE Security Functions Interface (TSFI) — A set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which TOE resources are accessed, mediated by the TSF, or information is obtained from the TSF.
TOE Security Policy (TSP) — A set of rules that regulate how assets are managed, protected and distributed within a TOE.
TOE security policy model — A structured representation of the security policy to be enforced by the TOE.
Trusted channel — A means by which a TSF and a remote trusted IT product can communicate with necessary confidence to support the TSP.
Trusted path — A means by which a user and a TSF can communicate with necessary confidence to support the TSP.
TSF data — Data created by and for the TOE that might affect the operation of the TOE.
TSF executable code – Includes any and all security relevant software and firmware.
TSF Scope of Control (TSC) — The set of interactions that can occur with or within a TOE and are subject to the rules of the TSP.
User — Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.
User data — Data created by and for the user that does not affect the operation of the TSF.