

# Protection Profile for Application Service Provider Operating System (ASP/OS PP)

*Jonathan S. Shapiro*  
Johns Hopkins University  
[shap@cs.jhu.edu](mailto:shap@cs.jhu.edu)

*David Chizmadia*  
Promia, Inc.  
[dchizmadia@promia.com](mailto:dchizmadia@promia.com)

Common Criteria Version: 2.1

**Draft Notice:** This protection profile is a work in progress. If it is broken, you get to keep all of the pieces you can locate.



# 1. Introduction

*Fill me in some day.*

## 2. TOE Description

### 2.1. General Description

An Application Service Provider (ASP) is an operator of computational facilities for contract customers, possibly provided on a rented or leased basis. ASPs are distinguished from conventional outsourcing operations by the fact that the software running on the rented/leased resources is at least partially dictated by the customer. That is, the operating basis for such systems is that the customer has paid for a standard arrangement of tools provided by the operator and may optionally augment those tools with software provided by the customer, a third party, or some combination thereof.

The operational requirements for an ASP include as subset cases:

1. Small-business servers, in which the operator is an agent of the user,
2. Large-business servers, in which the operator is an agent of the company that owns the equipment, but may not be entitled to access the content of any particular user,
3. Premises equipment in which the operator is a provider of some data-oriented service. This scenario has two further subcases:
  - A) Content delivery devices (including set top boxes) (such as multimedia content) in which the user is not generally offered the option to install software, or in which the enablement of available user options is accomplished through the intervention of the operator.
  - B) Enhanced premises devices (such as connected gaming devices), in which either the user may cause the installation of new applications and/or services that the user can subsequently cause to run.
4. Handheld devices, in which the user is the operator, not all software is equally trusted, and the users wish to protect sensitive or valuable information from software that later proves to be malicious, faulty, or both.

The common, characterizing attribute of these applications is that the computing platform is expected to run hostile software without being compromised or substantially degraded in performance.

An ASP Operating System (ASP OS) is an operating system capable of supporting the application service provider application and its subsets.

The stakeholders in an ASP OS fall into several categories:

1. The *operator*, who must meet obligations of specific performance to parties who have rented/leased service on a given compute platform. Simultaneously, the operator may wish for legal reasons to be distanced from liability resulting from the actions of users

and/or developers. This requires that the ASP OS enforce certain types of separations between these stakeholders.

2. The *user*, who has some application set they wish to run using the compute platform; presumably, but not necessarily, for the purpose of accomplishing some external goal.
3. *Developers* of software components and services installed by users or operators, who have warranties or contractual obligations that must be satisfied (contingent on operators, users, installers, and supplier developers meeting required preconditions). In addition to obligations of specific performance, a developer may optionally embody within their program content and algorithms that are proprietary, and therefore should not be disclosed to the user(s) of the system.
4. The *installer*, who configures the operating system on specific target hardware and potentially ensures that applicable tamper-proofing requirements are satisfied. The installer must inherently be trusted by the other parties.
5. The system architect, who architects, designs, and implements the ASP OS and its core runtime environment, and is inherently trusted by the other parties.

It is beyond the scope of this protection profile to specify means by which a secure bootstrap of the TOE is achieved prior to the execution of the first instruction of the TOE.

It is beyond the scope of this protection profile to specify how a developer may remotely determine whether the critical hardware tamper-proofing function of the installer has been satisfactorily met in the deployment of the TOE. This will eventually be the subject of a separate PP/ST combination.

## **2.2. Possible Threats**

There are three major attack categories to which an ASP OS is subject:

- **Escape:** Attempts by hostile software that is executing on the ASP OS to compromise their containment boundaries for the purpose of unauthorized disclosure.
- **Exhaustion:** Attempts to exhaust resources in order to impose starvation on other applications.
- **Penetration:** Attempts to bypass the normal mediation of software introduction in order to evade the security policies of the ASP OS. Alternatively, an attempt to compromise some element of the TSF.

## **3. Security Environment**

The following paragraphs explain in greater detail the Security Environment in which the ASP OS is expected to operate.

### **3.1. Assets**

The primary assets of concern to this PP are...

*This section really deals with rationale. Should it appear here, or in the rationale section?*

## **3.2. Assumptions**

### **A.GoodIPL**

An uncompromised installation and initial program load of the TOE has been performed.

In a case where the TOE implements multiple stages of initial program load, this assumption shall be taken to apply to the first of these stages. Once control of the bootstrap passes to the TOE, further assurance of correct execution are the responsibility of the TOE.

### **A.MediatedDisk**

Examination and modification of the persistent storage media used by the TOE is exclusively performed by the TOE and the bootstrap mechanism.

### **A.DiscreetIPL**

The bootstrap mechanism neither discloses nor modifies the content of the persistent storage media.

### **A.DedicatedHardware**

Following bootstrap, the TOE is the only code loaded on its host system, and receives control from the bootstrap in a well-defined and uncompromised state.

### **A.SafeHardware**

The hardware underlying the TOE provides no means by which non-privileged (in the eyes of the hardware) programs can execute hardware-privileged instructions without mediation by the hardware-privileged portion of the TOE.

### **A.TrustedHypervisor**

Any hypervisor present on the hardware underlying the TOE honors all of the previous assumptions.

### **A.Power**

Power to the TOE may be interrupted at any time.

### **A.RollBack**

In the event of power loss, it is permissible for the TOE to discard a bounded amount of progress in order to re-establish a well defined, correct security state.

### **A.Admin**

It is assumed that the TOE operator will make mistakes.

### **A.Network**

The TOE operates in a networked environment.

### **A.TrustedEnvironment**

No party shall improperly obtain authorized access to the TOE through non-technical means.

### **A.LimitationsOnDOS**

It is not within the scope of TOE assurance to limit consumption, control, or guarantee availability of resources not under control of the TOE.

In this context, the phrase “not under the control of the TOE” shall be taken to include TOE behavior that is mandated by *de facto* compliance with applicable standards. For example, TCP Syn-Flood attacks can be reduced only by strategic technical violations of the TCP protocol standard, and then only partially.

### **A.QuantumComputing**

The advent of quantum computing does not alter our understanding of computation in such a way as to invalidate the operational independence, non-simultaneity, and storage independence properties typically assumed of von Neuman computing devices.

This is a serious assumption. At the very least it has potential implications for selection of cryptographic techniques, as public key encryption based on prime factorization is known to be crackable using quantum computing techniques.

### **A.DiskECC**

The disk drive will implement a suitable error correcting code providing for the detection of multibit errors in modest number. The TSF can rely on this mechanism to detect decay in data written to the disk once it has been verified that the data was correctly written.

This is a serious assumption. At the very least it has potential implications for selection of cryptographic techniques, as public key encryption based on prime factorization is known to be crackable using quantum computing techniques.

## **3.3. Threats**

### **T.Resource**

An attacker (either internal or external) may attempt to overconsume resources so as to deny them to legitimate users.

### **T.BrainTransplant**

An attacker may exploit a flaw in the implementation of the TOE in order to cause that component to execute code determined by the attacker. In particular, this might include compromise of the TSF software.

### **T.ResourceAnalysis**

An attacker may exploit resource availability information (quota reporting) so as to discover information about a computation that is supposed to be discreet.

### **T.BadRollBack**

In the event of power loss, a rollback might occur in such a way that the TOE is not able to recover a well defined, correct security state.

### **T.OperatorError**

Errors made by the operator might leave the system security state compromised.

### **T.BadPolicy**

Through error or malice, an operator may put in place a system security policy that is incorrect or unenforceable.

#### **T.ProprietaryDisclosure**

The proprietary state of a program may be disclosed through a failure in the design or application of the TOE's security policies and mechanisms.

#### **T.Leakage**

The private state of a program may be disclosed by a serving component through a failure of confinement on that component.

#### **T.Masquerade**

An attacker may seek to defeat the TOE's authentication mechanisms in order to consume the resources of (masquerade as) a legitimate user.

#### **T.OperatorMalice**

A hostile operator may seek to examine the content of protected state owned by a user.

#### **T.OperatorLiability**

An operator may be exposed to liability and damages resulting from malicious or illegal actions.

#### **T.BadInstall**

An operator may seek to introduce compromised services into the system by altering the content of the installation media.

#### **T.Covert**

Two parties not having an authorized path of communication may seek to transmit data using covert channels of communication.

#### **T.Abuse**

A hostile program may use its authority in improper ways.

### **3.4. Security Policies**

#### **P.BoundedAssurance**

Assurance in the TOE is bounded by the assurance with which the Operating Environment assumptions have been satisfied.

#### **P.OperatorExposure**

The operator shall not be exposed to the content of TOE users except where the operator *is* the user.

#### **P.AdminTools**

The administrative tools shall restrict the administrator to (de)allocation of system resources and monitoring the consumption of such resource.

#### **P.Authentication**

All nontrivial use of the TOE shall be mediated by authentication.

The degree of authentication required in the TOE itself may be mitigated by the target application. For example, an embedded premises device might plausibly rely on the physical protection of the premises as the principal means of authentication for users with direct physical access to the device.

Note that the assurance of the TOE is bounded by the assurance of its authentication mechanism.

### **P. Anonymous Interface**

The P. Authentication interface notwithstanding, the TOE may permit unauthenticated access to such facilities as may be necessary to perform authentication or (if unmediated account creation is permitted) establish a new identity for purposes of TOE authentication.

### **P. Well Defined State**

After IPL, the TOE shall always initiate execution from a well-defined security state.

### **P. Policy Rigor**

The TOE shall not allege to enforce a security policy unless (a) it can be shown that there exists some means by which the policy is in principle enforceable given the protection mechanisms implemented by the TOE, and (b) the TOE in fact implements some embodiment that performs such enforcement using these mechanisms.

### **P. Least Authority**

A program executing under the TOE shall have the least authority sufficient to perform its function.

A corollary to this is that components executing under the TOE shall have no inherent authority at the moment of their creation.

### **P. Resource**

A program executing under the TOE shall have no inherent ability to allocate resources.

A corollary to this is that the resources authorized to a program must be provided by its instantiator. In the case of components corresponding to user sessions, the instantiator may be software operating as an agent of the system operator.

### **P. Confinement**

Every untrusted program executing under the TOE shall execute within a confinement boundary.

### **P. Trust**

The assurance of any TOE-supplied operating environment shall be bounded by the assurance with which its trusted, unconfined subsystems comply with the policies and requirements stated in this protection provilde.

### **P. Suspicion**

The TOE shall treat all code outside the TOE as potentially hostile.

### **P. RsrcAuth**

No program shall allocate or use a resource for which it lacks authorization.

### **P.RsrcMux**

The TOE shall reify interfaces for the exact control of resource multiplexing.

### **P.Conservation**

The TOE shall be resource-conservative.

The TOE operates from a finite pool of resources. It allocates and mediates access to these resources, but neither creates nor destroys them. The correctness of the TOE's execution must not rely on the assumption that available resources are unbounded.

### **P.CommonCarrier**

The TOE shall be designed in such a way as to facilitate operation under "common carrier" status, where applicable.

The "Common Carrier" statute provides for reduction or elimination of operator liability where it can be shown that the operator lacked technical means to know or control the activities of users. The TOE should be designed in such a way as to preserve this separation so that the operator has the option of invoking this statutory protection where it applies.

### **P.AccXferConsent**

The TOE shall be designed in such a way as to ensure that no user can alter the access rights of a second user without the explicit consent of the recipient.

The "Common Carrier" statute provides for reduction or elimination of operator liability where it can be shown that the operator lacked technical means to know or control the activities of users. The TOE should be designed in such a way as to preserve this separation so that the operator has the option of invoking this statutory protection where it applies.

## **4. Security Objectives**

### **4.1. Security Objectives for the TOE**

#### **SO.NoSuperUser**

Operators shall not have visibility into the operation and state of software running on behalf of users.

**Reference:** P.OperatorExposure

**Reference:** P.CommonCarrier

#### **SO.SeparationOfAllocation**

The ability to authorize the allocation of resources shall not imply the ability to examine the content of the resources allocated.

**Reference:** P.OperatorExposure

**Reference:** P.CommonCarrier

### **SO.FailSecure**

The TOE must be designed such that it is always able to recover a well-defined system state following a power failure or crash.

**Reference:** A.Power

**Reference:** P.WellDefinedState

### **SO.Interfaces**

Code shall only interact with other code (including the TOE) through well-defined interfaces.

### **SO.IfEncap**

The TOE shall enforce encapsulation of the interfaces by which components communicate.

### **SO.IfSafe**

Every operation on each interface exported by the TOE shall transform the state of the TOE from one well-defined security state to another well-defined security state.

**Reference:** A.Power

**Reference:** P.WellDefinedState

### **SO.Availability**

The TOE shall provide means by which the operator can establish minimal resource availability commitments. The TOE shall ensure that the resources bound under such commitments are always available for use by authorized programs.

### **SO.Quotas**

The TOE shall provide means by which the operator can establish maximal bounds on each resource availability commitment. The TOE shall ensure that the resources provided under each resource commitment do not exceed the bound on those authorized.

### **SO.Progress**

The TOE shall ensure that any program executing under the TOE shall continue to make progress so long as sufficient resources have been committed for its use. The TOE shall further ensure that the preconditions for progress are well-defined.

### **SO.LibFuns**

All core TOE library functions that read or write buffers shall include in their interface a specification of the buffer length, and shall have well-defined behavior in the event that this limit is exceeded. All TOE components shall be built exclusively using core TOE libraries.

### **SO.Fuzziness**

Where feasible, the TOE resource management interfaces shall be designed to minimize the exposure of global resource availability information. Where this is not feasible, the TOE shall apply suitable countermeasures (including fuzzy reporting) in reporting these figures as a covert channel defense.

### **SO.Restart**

The TOE shall ensure that at all times there is a well-defined security state from which to restart after power loss, and that the interval between the execution of the first TOE instruction and the first user application instruction preserves this security.

### **SO.Confinement**

The TOE shall provide a mechanism by which a program can be prohibited from communicating via unauthorized channels.

### **SO.Authentication**

The TOE, with support from its environment, will ensure that each user is uniquely identified, and that the claimed identity is authenticated, before the user is granted access to the TOE facilities.

### **SO.NonDisclosure**

The TOE shall be so designed that a given user's content is protected from disclosure to any other party without the consent of some program running on behalf of the user. For purposes of this objective, the system operator is specifically included in the list of "other parties."

### **SO.PositiveAgreement**

The TOE shall be designed to ensure that no user can make information available to another party unless the receiving party has agreed to accept that information.

The issue here is that in some systems I can alter the permissions on an object in such a way as to grant you access to the object, even though you have never heard of me. This lends itself to both liability issues and enticement errors.

### **SO.AssuredInstall**

The TOE shall provide appropriate measures to confirm that code installed by any party has not been altered by the operator.

### **SO.MinimalAuthority**

The TOE shall be designed in such a way that each of its components holds the minimal authority needed to perform its function. The TOE shall facilitate in its interfaces and design the construction of application software that likewise satisfies this objective.

Trigger locks just ensure that gene pool pruning is suboptimal, but guns should be unloadable.

### **SO.SoundAdministration**

The policy administration tools of the TOE shall only permit the construction of consistent and enforceable security policies.

### **SO.NoDetect**

The TOE shall *not* provide any means by which the operator might detect the erroneous or malicious behavior of validly admitted programs.

Such detection violates confinement. There is a need for the operator to be able to observe aggregate conditions on the machine. For example, the operator needs to be able to learn both (a) total available space with which to create accounts, and (b) amount of space currently under the control of a given user. The administrator should *not* be able to learn in any fine grain way the fraction of authorized space that has been allocated.

Analogous issues arise with most security detection mechanisms. In general, the objective of this PP is to close security holes by making improper actions impossible *a priori* rather than providing *post hoc* mechanisms for discovery.

A notable exception to this rule concerns the behavior of events *outside* the scope of legitimate user activity. For example, the system aggregate number of packets sent and received, and any information voluntarily logged by the networking subsystem with the consent of a user, is valid content for an administrator to observe and act upon.

## **SO.PktFilter**

The TOE shall provide means by which the operator can filter the machine's interface to the outside, networked world. This means shall *not* provide the operator with authority to examine network traffic content.

## **4.2. Security Objectives for the Environment**

## **4.3. Conclusions and Security Objectives Rationale**

# **5. Security Requirements**

## **5.1. Functional Requirements**

### **5.1.1. Access Control SFP**

The requirements of the ASP OS Access Control SFP are captured below under the ASFP functional requirements. The following text motivates both their inclusion and the limitations on the constraints that they impose.

The underlying real-world operating model of an ASP OS is that the user has leased resources and is entitled to do with them whatever they wish, subject only to the limitations of applicable law, turing-equivalence, the applications they choose to run, and the obligations of the ASP OS to support other users without disruption.

### **Mandatory Policies**

Given the ASP scenario, the sole mandatory policy enforced by the ASP OS is the P.AccXferConsent policy above. This is addressed by the Access Control SFP functional requirements below.

## ***Discretionary Policy***

In general, it is therefore inappropriate for an ASP OS to impose a mandatory access control policy. Individual applications may impose restrictions on the services that they offer to their user,<sup>1</sup> but such restrictions lie outside the scope of the ASP OS or the TSF.

Further, users are entitled to communicate with each other, provided that both parties consent to the communication. The communication by which such consent is established necessarily occurs outside the scope of the TSF. The responsibility of the TSF in this regard is solely to provide such support for rendezvous as may be required in order for a user who *wishes* to use secure means of communication to establish such rendezvous with adequate provision for security and privacy.

## ***Separation of Operator Role***

That said, it is appropriate for the ASP OS to defend itself from actions taken by one user that might impede the satisfaction of the TSF contract with some other user. In order to do this, the TSF must maintain a complete separation of the operator role from the user role.

As there are a number of ways to implement this separation of roles, this protection profile does not specify a particular method of separation. Satisfactory methods might include, but are not limited to, session labels or compartmentalization of sessions into roles by prior design.

It should be noted that given the strong policy of separation between operator and user, and between user and user required to establish and sustain common carrier status, many of the cases in conventional systems where it is useful to temporarily change roles to an administrator are unhelpful or precluded in an ASP OS.

## **5.1.2. Information Flow SFP**

The requirements of the ASP OS Information Control SFP are captured below under the IFSFP functional requirements. The following text motivates both their inclusion and the limitations on the constraints that they impose.

A secure ASP OS platform in isolation is relatively useless. To be effective, the ASP OS platform must provide a runtime environment that facilitates the creation of subsystems that provide information, access right, and resource isolation.

As a general-purpose secure computing platform, an ASP OS should facilitate the creation of nested secure operating environments that may internally implement their own mandatory and discretionary policies. Nesting such environments requires some degree of support from the underlying abstract machine provided by the ASP OS. In particular, it is necessary that the protection primitives provided by the ASP OS be sufficiently powerful to implement the confinement policy.

The overall goal of the ASP OS information flow policy is to achieve the following (non normative) objectives:

---

<sup>1</sup> For example, a set-top box may validate that the box remains subscribed before presenting a paid movie channel.

1. To the greatest extent possible, all applications and services should be instantiated as confined protection domains that are initially under the exclusive control of their requestor (i.e. the subject who requested their instantiation).
2. To the greatest extent possible, interaction between any subsystem S and any other mutable subjects or objects that are not exclusively controlled by S shall be performed through a mediator trusted by the user.
3. Distinct sessions, whether operating on behalf of the same user or distinct users, should be mutually isolated by the TSF, and all establishment of initial communication channels between these sessions should be mediated by TSF software.

Both the first and second goals are sensible design objectives in a secure system. Regrettably, neither can be reduced to a universally enforceable information flow requirement, because both rely on the correct design of applications, and in particular on the careful design of application shells (which can and should be the subject of a separate protection profile).

### 5.1.3. Requirements

#### ASFP.ExplicitAccess

Any time a subject performs any operation on any interface, there must exist an access control rule that either (a) authorizes this action or (b) prohibits it. An access control mechanism in which unauthorized actions are precluded – that is, in which all actions that can be articulated by a subject are by definition authorized – sufficiently satisfies this requirement.

This implies that the TSF shall logically maintain, at a minimum, a continuously up to date set containing tuples of the form (subject id, object id, interface id, operation id), and optionally other attributes, that shall be used as the basis for performing the access predicate test. This set shall be referred to as the **Access Matrix**.

This note should not be taken to require that the TSF explicitly implement such a matrix; only that its operational access control decision procedure can be expressed in terms of membership tests against and manipulation of such a matrix.

#### ASFP.XferMediation

When a subject performs any operation whose effect is to increase the authorities of a second subject operating as an agent of a different user, the TSF shall ensure that at least one of the following conditions holds:

1. The authority transfer is permitted under some chain of authorization previously established between the subjects, or
2. The authority transfer is mediated by the TSF in such a way as to allow the recipient user to compartmentalize this new authority until the desirability of its acceptance can be determined by the recipient.

### **ASFP.RoleSep**

The TSF shall be architected, designed, and implemented in such a way as to ensure that system administrative actions may only be undertaken in an operator role, and that there shall exist no means by which a session operating in one role can alter its current role.

### **IFSFP.Rigor**

The TSF shall (allege to) implement only such information flow policies as can be rigorously shown to be enforceable given the primitive protection mechanisms of the TSF.

### **IFSFP.Confinement**

The TSF shall provide a means of confining newly instantiated subsystems.

### **IFSFP.ConfineDefault**

The default means of program instantiation provided by the TSF shall instantiate programs as confined subsystems that are initially accessible exclusively by their instantiating (requesting) subject.

### **IFSFP.ConfineCheck**

The TSF shall provide means by which a subject may reliably determine, prior to the instantiation of a new subsystem, whether the subsystem shall have any unauthorized channels of communication.

### **IFSFP.Flows.1**

A subject shall be able to transfer authority according to the access control policies dictated by the access control SPF. It follows from this that the permissible information flows in the TSF are those resulting from the transitive, reflexive, and (in subject-subject communications) symmetric closure of the stepwise information flows authorized by the access control SPF.

### **IFSFP.Flows.2**

Where the TSF is required to restrict such flows, it is the responsibility of the TSF to enforce isolated compartments and mediate interactions between these compartments using a mediator that enforces the desired restriction(s).

### **SFR.UnitsOfOperation**

The execution of the TOE shall be conceptually divisible into atomic units of operation, each of which shall transform the state of the TOE from one well-defined state into another well-defined state.

**Reference:** A.Power

**Reference:** P.WellDefinedState

### **SFR SFR.UserExecutionModel**

The execution of any user-mode instruction shall be expressible in the TOE security model as a sequence of TOE units of operation.

**Reference:** A.Power

**Reference:** P.WellDefinedState

**Reference:** SFR.UnitsOfOperation

### **FDP\_ACC.2.1**

The TSF shall enforce the [access control SFP](#) on all subjects and objects and all operations among subjects and objects covered by the SFP.

### **FDP\_ACC.2.2**

The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

**Reference:** FDP\_ACF.1

### **SFR FDP\_ACF.1.1**

The TSF shall enforce the [access control SFP](#) to objects based on [subject identity](#), [object identity](#), [interface identity](#), and [operation identity](#).

### **FDP\_ACF.1.2**

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [\(logical\) membership of the appropriate access tuple in the access matrix](#).

### **FDP\_ACF.1.3**

The TSF shall explicitly authorise access of subjects to objects based on the following additional rules:

1. Those portions of the TSF that are implemented as isolated subjects may be granted such minimal initial authority as is needed to perform their designed function within the TSF. This assignment may be performed as a special case at initialization-time, or by prior construction.
2. Notwithstanding [ASFP.XferMediation](#), the TSF authentication agent(s) shall be permitted access to grant rescindable access to the session connection transport to subject(s) acting on behalf of a newly authenticated user.

### **FDP\_ACF.1.4**

The TSF shall explicitly deny access of subjects to objects based on the following rules:

1. Following a failure of the TSF, the TSF shall take measures to ensure that all subject authority over external communication channels that cannot be reauthenticated by the TSF shall be rescinded.
2. Notwithstanding (1), those subjects that are part of the TSF that perform authentication functions may retain or re-establish authority over unauthenticated external communication channels.

In English: If the machine crashes, you probably have to log in again.

**Depends On:** FDP\_ACC.1

**Depends On:** FMT\_MSA.3

### **FDP\_ETC.1.1**

The TSF shall enforce the access control SFP and information flow control SFP when exporting user data, controlled under the SFP, outside of the TSC.

### **FDP\_ETC.1.2**

The TSF shall export the user data without the user data's associated security attributes.

The only plausible security attribute in the ASP OS scenario is the fact that the user has access to the information being exported. While individual applications (e.g. content protection software) may impose additional restrictions, these are outside the scope of the ASP OS *per se*, except to the extent that the ASP OS must facilitate marking by applications by ensuring that the applications are properly isolated from tampering once installed.

**Depends On:** FDP\_ACC.1

**Depends On:** FDP\_IFC.1

### **FDP\_IFC.2.1**

The TSF shall enforce the **information flow control SFP** on **all subjects, information, and access rights** and all operations that cause that information to flow to and from subjects covered by the SFP.

### **FDP\_IFC.2.2**

The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

**Depends On:** FDP\_IFF.1

### **FDP\_IFF.2.1**

The TSF shall enforce the **information flow control SFP** based on the following types of subject and information security attributes: **all attributes recorded in the access matrix**.

### **FDP\_IFF.2.2**

The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules, based on the ordering relationships between security attributes hold: **specified by the information flow SFP and associated information flow-related requirements (IFSFP.\*).**

### **FDP\_IFF.2.3**

The TSF shall enforce any **additional information flow control rules** that are necessary to **enforce the ASFP.RoleSep** requirement.

### **FDP\_IFF.2.4**

The TSF shall provide **no additional SFP capabilities**.

### **FDP\_IFF.2.5**

The TSF shall explicitly authorise an information flow based on the following rules: **(none)**.

All necessary additional information flow authorizations are implied by the combination of the additional access control authorizations under FDP\_ACF.1.3 and the previously permitted information flows.

#### **FDP\_IFF.2.6**

The TSF shall explicitly deny an information flow based on the following rules: *(none)*.

All necessary additional information flow restrictions are implied by combination of the additional access control restrictions under FDP\_ACF.1.4 and the previously permitted information flows.

#### **FDP\_IFF.2.7**

The TSF shall enforce the following relationships for any two valid information flow control security attributes:

1. There exists an ordering function that, given two valid security attributes, determines if the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable; and
2. There exists a "least upper bound" in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is greater than or equal to the two valid security attributes; and
3. There exists a "greatest lower bound" in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is not greater than the two valid security attributes.

*I am not convinced that this formulation makes sense here.*

**Depends On:** FDP\_IFC.1

**Depends On:** FMT\_MSA.3

*Fix: I have removed FDP\_IFF.3, because it is not clear at this point how to specify it correctly, and it's inclusion can be deferred.*

#### **FDP\_ITC.1.1**

The TSF shall enforce the [access control SFP](#) and [information flow control SFP](#) when importing user data, controlled under the SFP, from outside of the TSC.

#### **SFR FDP\_ITC.1.2**

The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

#### **SFR FDP\_ITC.1.3**

The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: *(none)*.

**Depends On:** FDP\_ACC.1

**Depends On:** FDP\_IFC.1

**Depends On:** FMT\_MSA.3

### **FDP\_RIP.2.1**

The TSF shall ensure that any previous information content of a resource is made unavailable upon the deallocation of the resource from all objects.

### **FDP\_SDI.2.1**

The TSF shall monitor user data stored within the TSC for data corruption on all objects, based on the following attributes:

1. When storing data to persistent storage, the TSF shall make use of the underlying error detection/correction mechanisms of the disk, and will detect and report failures on reread.
2. Where a particular persistent storage device does not innately provide an effective correction facility, the TSF shall store data in such a way as to independently compute and validate an appropriate error detection check.
3. When storing data in volatile memory that is not mutable or has not yet been mutated, the TSF shall compute an appropriate error detecting code to detect erroneous modification of such data due to ambient radiation.
4. When storing data in volatile memory that is mutable, the TSF shall maintain an appropriate error detecting code on all data structures containing or caching information corresponding to entries in the access control SFP access matrix.
5. When writing mutated in-memory data to persistent storage, the TSF shall take measures to immediately mark the in-memory copy unmodified, and enforce all checks required above until such time as this object has again been mutated.

### **FDP\_SDI.2.2**

Upon detection of a user data integrity error, the TSF shall either:

1. Report this error as an exceptional return value to the operation that caused the integrity error to be discovered by the TSF, or.
2. Report this error by delivering a software-synthesized memory exception to the appropriate memory exception handler. In meeting this requirement, The TSF must ensure that the responsible exception handling agent for any user memory region is always well defined.

### **FIA\_AFL.1.1**

The TSF shall detect when 3 unsuccessful authentication attempts occur related to login on devices other than the physical system console.

### **FIA\_AFL.1.2**

When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall: complain bitterly.

1. Disable further authentication attempts by this user on devices other than the physical system console. Brute force attacks on the console should be further discouraged by imposing a delay between successive login attempts.

2. Attempt on a best-effort basis to notify the user using a previously established electronic mail address that their account has been disabled due to repeated authentication failure.
3. Provide means by which the system operator can re-enable the account after causing a replacement password to be generated.

The re-enablement needs to be thought about carefully, because injudicious use of it could allow the operator to obtain access to the user's account.

A better design might be to facilitate a fallback challenge-response based re-authentication mechanism, but this has implications for privacy management.

**Depends On:** FIA\_UAU.1

### **FIA\_ATD.1.1**

The TSF shall maintain the following list of security attributes belonging to individual users:

1. The role (operator or user) associated with this principal.
2. The authentication mechanism(s) required for this principal to authenticate.
3. Whether login for this principal is currently permitted on any device other than the system console.

### **FIA\_SOS.1.1**

The TSF shall provide a mechanism to verify that secrets meet the following quality metrics:

1. For passwords: not a dictionary word, contains at least one numeric and one non-alphanumeric character, does not match any information in the per-user description (such as their real name or account name). Does not match the set of strings generated by a specified set of transformations on any of the preceding sources of inappropriate passwords.
2. For cryptographic keys: (a) that they satisfy a minimum key length of 1024 bits, (b) ?? that they are endorsed by a trusted certificate authority, possibly including the TSF itself.

It would be interesting to explore the use of probabilistic filtering here. If we can do part of speech with high confidence, we can surely do "is this a plausible word in a plausible language"

### **FIA\_SOS.2.1**

The TSF shall provide a mechanism to generate secrets that meet the quality metrics specified in FIA\_SOS.1.1. In support of this, the TSF must support a good source of entropy for cryptographic key generation.

It is not clear to me how to specify how to recognize a good enough source of entropy. Peter Gutman should know.

## **FIA\_SOS.2.2**

The TSF shall be able to enforce the use of TSF generated secrets for [session keys used in cryptographically protected sessions](#).

## **FIA\_UAU.2.1**

The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

**Depends On:** FIA\_UID.1

## **FIA\_UAU.3.1**

The TSF shall **prevent** use of authentication data that has been forged by any user of the TSF.

There is a definitional problem here in the question of what constitutes a forgery. The above selections apply to conventionally encrypted password authentication data. For cryptographic authentication keys, it is within the intended behavior of the ASP OS to allow one principal to extend authority to another -- indeed, we cannot prevent this. Therefore, for cryptographic-based authentication our test will be limited to validating that the authentication key was introduced into an account by a currently authorized user of that account.

In connection with this, it may be desirable to distinguish cryptographic keys that can authorize other cryptographic keys and those that cannot. This does not preclude proxying, but it does limit the problem of rescinding access to a potentially smaller number of cryptographic keys, which is important because the proposed key introduction mechanism does not preserve chains of authorization.

## **FIA\_UAU.3.2**

The TSF shall **prevent** use of authentication data that has been copied from any other user of the TSF.

See comments above under FIA.3.1. Copying is just a special case of forgery.

## **FIA\_UAU.4.1**

The TSF shall prevent reuse of authentication data related to [most recent three passwords](#).

## **FIA\_UAU.5.1**

The TSF shall provide [password-based and/or cryptographic key based](#) to support user authentication. [As a special case, the TSF may designate a particular hardwired interaction device to be always authenticated to some particular user.](#)

## **FIA\_UAU.5.2**

The TSF shall authenticate any user's claimed identity according to [at least one of the mechanisms specified in FIA\\_UAU.5.1](#).

## **FIA\_UAU.6.1**

The TSF shall re-authenticate the user [whenever the user wishes to alter their password, or when a failure of the TSF or a disconnection of an external communication line implies that the user associated with the channel can no longer be known with certainty.](#)

### **FIA\_UAU.7.1**

The TSF shall provide only prompts for each needed piece of information and a final notice of success/failure to the user while the authentication is in progress.

**Depends On:** FIA\_UAU.1

### **FIA\_UID.2.1**

The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

### **FIA\_USB.1.1**

The TSF shall associate the appropriate user security attributes with subjects acting on behalf of that user.

This association requirement shall be satisfied by a system designed in such a way that subjects, by prior design, cannot violate the constraints that would be imposed by those security attributes.

**Depends On:** FIA\_ATD.1

### **FMT\_MOF.1.1**

The TSF shall restrict the ability to compose, disable, or enable all security management functions to the authorized operator, subject to the restriction that the management interface will not permit alterations to the security management functions or their application that would violate the requirements otherwise specified in this protection profile. The TSF shall restrict the ability to modify the behaviour of the security management functions, and the rules for proper composition of those functions, to the system architect.

**Depends On:** FMT\_SMR.1

### **FMT\_MSA.1.1**

The TSF shall ensure that security attributes are uniquely associated with each role, and that all roles are disjoint. The TSF shall restrict the ability to alter passwords and cryptographic authentication keys associated with an account to the authorized user of that account. The TSF shall permit the operator to add or delete new accounts, and to alter the resource commitments to each account, but shall otherwise prevent the modification of account security attributes.

**Depends On:** FDP\_ACC.1

**Depends On:** FDP\_IFC.1

**Depends On:** FMT\_SMR.1

### **FMT\_MSA.2.1**

The TSF shall ensure that only secure values are accepted for security attributes.

**Depends On:** ADV\_SPM.1

**Depends On:** FDP\_ACC.1

**Depends On:** FDP\_IFC.1

**Depends On:** FMT\_MSA.1

**Depends On:** FMT\_SMR.1

### **FMT\_MSA.3.1**

The TSF shall ensure by construction that appropriate default values for security attributes are used to enforce the SFP.

### **FMT\_MSA.3.2**

The TSF shall **not** allow the **specification of alternative initial values** to override the default values when an object or information is created.

**Depends On:** FMT\_MSA.1

**Depends On:** FMT\_SMR.1

### **FMT\_MTD.1.1**

The TSF shall restrict the ability to **query, modify, delete, clear,** or replace the TSF initial system image to the operator, subject to validity check by the TSF.

**Depends On:** FMT\_SMR.1

### **FMT\_MTD.2.1**

The TSF shall restrict the specification of the limits for **audit log size** to the operator.

### **FMT\_MTD.2.2**

The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: **the audit log shall be reused in circular fashion. In no case shall the TSF permit the exhaustion of a TSF limit to result in a cessation of TSF operation or the failure of any TSF security requirement.**

**Depends On:** FMT\_MTD.1

**Depends On:** FMT\_SMR.1

### **FMT\_MTD.3.1**

The TSF shall ensure that only secure values are accepted for TSF data.

**Depends On:** ADV\_SPM.1

**Depends On:** FMT\_MTD.1

### **FMT\_REV.1.1**

The TSF shall restrict the ability to revoke security attributes associated with the **users** within the TSC to **the operator.**

### **FMT\_REV.1.2**

The TSF shall enforce the rule **that revocation shall only occur as the result of account deletion, reduction of allocatable resource allocated to a contract, or disabling of an account.**

**Depends On:** FMT\_SMR.1

### **FMT\_SAE.1.1**

The TSF shall restrict the capability to specify an expiration time for passwords to the operator, subject to constraints imposed by the TSF, and to the user, subject to the constraint that no user-established expiration time shall be longer than that imposed by the most restrictive of the operator and the TSF.

### **FMT\_SAE.1.2**

For each of these security attributes, the TSF shall be able to disable the associated account after the expiration time for the indicated security attribute has passed.

**Depends On:** FMT\_SMR.1

**Depends On:** FMT\_STM.1

### **FMT\_SMR.2.1**

The TSF shall maintain the roles: normal user, operator.

### **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 the normal user and operator roles are restricted to disjoint sessions are satisfied.

**Depends On:** FIA\_UID.1

### **FMT\_SMR.3**

#### **FMT\_SMR.3.1**

The TSF shall require an explicit request to assume the following roles: all. The explicit request shall take the form of session initiation or session switching via the session manager.

**Depends On:** FMT\_SMR.1

### **FPR\_ANO.2**

**Supersedes:** FPR\_ANO.1

#### **FPR\_ANO.2.1**

The TSF shall ensure that normal users are unable to determine the real user name bound to operations, subjects, or objects.

#### **FPR\_ANO.2.2**

The TSF shall provide all services to all subjects without soliciting any reference to the real user name.

### **SFR\_PSE.1**

The TSF shall ensure that users and subjects are unable to determine the real user name bound to any subject, operations, or objects.

### **FPR\_UNL.1.1**

The TSF shall ensure that users and/or subjects are unable to determine whether any and all operations were caused by the same user, or are related as follows in any way, except that a serving object/subject may determine whether two requests arrived on the same interface.

### **SFR.UNO.1**

The TSF shall ensure that users and subject are unable to observe any operations on an object by any other user or subject, unless (a) both parties have authorized access to a common object and (b) the behavior implemented by that object implements some other policy.

### **FPT\_AMT.1.1**

The TSF shall run a suite of tests during initial start-up, periodically during normal operation, and at the request of the operator to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

### **FPT\_FLS.1.1**

The TSF shall preserve a secure state when the following types of failures occur: (all software failures).

**Depends On:** ADV\_SPM.1

### **FPT\_RCV.3.1**

When automated recovery from a failure or service discontinuity is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

### **FPT\_RCV.3.2**

For all software failures, for power losses, and for planned shutdowns, 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 5 minutes of total state lost 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. For subjects and objects created sufficiently close to the time of failure, knowledge of the *existence* of these objects may be lost. It is not required that the non-recovery of these objects be determinable.

**Depends On:** FPT\_TST.1

**Depends On:** AGD\_ADM.1

**Depends On:** ADV\_SPM.1

#### **FPT\_RCV.4.1**

The TSF shall ensure that for all failure scenarios resulting from software error, power loss, or planned shutdown, all security functions have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

**Depends On:** ADV\_SPM.1

#### **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.

For TSP enforcement predicates that are universally deterministic functions, the TSF may cache the prior results of these functions, provided it ensures that this cache is properly invalidated when the values from which the predicate result was computed (may) have changed.

#### **FPT\_SEP.3.1**

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.3.2**

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

#### **FPT\_SEP.3.3**

The TSF shall maintain the part of the TSF that enforces the access control and/or information flow control SFPs in a security domain for its own execution that protects them from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to the TSP.

#### **FPT\_STM.1.1**

The TSF shall be able to provide reliable time stamps for its own use.

#### **FPT\_TST.1.1**

The TSF shall run a suite of self tests during initial start-up, periodically during normal operation, at the request of the operator, and prior to committing any transaction recording security-sensitive state to demonstrate the correct operation of the TSF.

#### **FPT\_TST.1.2**

The TSF shall provide authorised users with the capability to verify the integrity of TSF data.

#### **FPT\_TST.1.3**

The TSF shall provide authorised users with the capability to verify the integrity of stored TSF executable code.

**Depends On:** FPT\_AMT.1

## **FRU\_FLT.2**

**Supersedes:** FRU\_FLT.1

### **FRU\_FLT.2.1**

The TSF shall ensure the operation of all the TOE's capabilities when the following failures occur: **all software failures, power loss, planned shutdown.**

**Depends On:** FPT\_FLS.1

## **SFR.SCHED.1**

The TSF shall provide a resource usage control mechanism that ensures the following guarantees:

1. All access to all resources shall be subjected to the access control SFP.
2. No action taken by any user shall cause the minimum guarantee of resource associated with a given resource pool to be violated.
3. The system operator shall always have a guarantee of sufficient resource to perform orderly emergency system administration function.

## **SFR.RSA.1.1**

The TSF shall establish resource pools for all resources, and shall ensure that all allocation of resource by a subject occurs via some resource pool which that subject possesses the authority to consume.

### **SFR.RSA.1.2**

For fungible resources, a resource pool shall have the ability to restrict both the minimum and maximum amount of that resource that a subject possessing the authority to utilize that pool may simultaneously consume.

### **SFR.RSA.1.2**

For renewable resources, a resource pool shall have the ability to restrict provide a lower bound guarantee and an upper bound restriction on the amount of such resource that a subject possessing the authority to utilize that pool may consume over a defined period of time.

### **SFR.RSA.2.1**

The TSF shall ensure that the pools allocated to each user account shall always be defined with both minimal and maximal resource bounds.

### **SFR.RSA.3.1**

The TSF shall ensure that the minimum resource guarantee of a resource pool shall always be satisfiable by the TSF.

## **FTP\_TRP.1.1**

The TSF shall provide a communication path between itself and **local and remote** users that is logically distinct from other communication paths and provides assured

identification of its end points and protection of the communicated data from modification or disclosure.

#### **FTP\_TRP.1.2**

The TSF shall permit **the TSF and local users** to initiate communication via the trusted path.

#### **FTP\_TRP.1.3**

The TSF shall require the use of the trusted path for **initial user authentication, and for use of any application service in which interactive user confirmation is required.**

An example of this might include the open/save file dialog box in a GUI system.

### **5.2. Assurance Requirements**

*These need to be filled in too someday.*