Separation Kernel Protection Profile Revisited: Choices and Rationale

Timothy E. Levin, Thuy D. Nguyen, Cynthia E. Irvine
Naval Postgraduate School
Michael McEvilley
The MITRE Corporation
{levin, irvine, tdnguyen}@nps.edu, mcevilley@mitre.org

Abstract: *A variety of critical decisions were made during the development of the* U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness *(the SKPP); in addition several errata that have come to light since its publication. This paper is intended to help future SKPP users to better understand the intent of the requirements.*

1. Introduction

The U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness (or "SKPP"), is and has been the basis for the development and evaluation of various high assurance products. Every protection profile is required to provide reasoning and explanations regarding its requirements. However, the general case is that these ad hoc explanations and structured rationale are often brief, pro forma statements that do not provide extensive background due to resource limitations during protection profile development. The purpose of this paper is to explain in more depth of a variety of critical decisions in the development of the SKPP, and describe several errata that have come to light since its publication. This will help future users of the SKPP to better understand the intent of the requirements.

Several challenges during development of the SKPP resulted in its four-year project lifetime. Primarily, the project explored several novel areas in Common Criteria requirements specification, some of which are discussed in the rest of this paper. It was the first high robustness product protection profile sanctioned by the U.S. Government. It addressed new requirements in the specification of Target of Evaluation (TOE) hardware. It developed a new abstraction and related requirements for least privilege in separation kernels, and a means by which a TOE could ensure the adequacy of its trusted subjects (with respect to the basic abstract policy). It developed the *configuration vector* abstraction by which the basic abstract policy and other requirements for configuration could be provided to the TOE. It developed specifications for a configuration tool to be used by security administrators to prepare *configuration vectors* for submission to the TOE. And finally, the authoring team developed concepts and requirements for dynamic, runtime changes to the security configuration of the TOE.

Development of the SKPP began in 2003 with a series of planning meetings called by the National Security Agency (NSA). The NSA then assigned a small group of security analysts from The MITRE Corporation and the Naval Postgraduate School (NPS) to build a U.S. Government High Robustness protection profile for separation kernels, using an existing, unpublished Partitioning Kernel Protection Profile [PKPP] skeleton[1] as "a starting point." The members of this authoring team (who are also the authors of this paper) were: Michael McEvilley from The MITRE Corporation; and Cynthia Irvine, Thuy Nguyen, and Timothy Levin from NPS. Hugo Badillo, NSA, managed the project; and Cynthia Irvine provided authoring team leadership. This team brought to the project a broad collective background, including extensive experience with authoring Common Criteria Protection Profiles; high assurance security kernel development; formal verification; product evaluation; and engineering of operational requirements that in part motivated the need for the SKPP.

[1] The PKPP was developed by members of The Open Group Real-Time and Embedded Systems Forum

2. Separation Kernel

A separation kernel is an executive that serves a simple purpose: it controls all of the physical resources of a system (i.e., hardware, firmware, software); it constructs different types of resource abstractions from the physical resources, while leaving other physical resources in their original form; it *exports* a subset of these resources; it allocates the exported resources to partitions; and it controls information flow between the partitions and between exported resources. The exported resources, partitions, and allowed flows are specified in a *configuration-vector* input to the separation kernel during its initialization.

A subset of the exported resources are active, i.e., they can cause operations to be performed, and are commonly referred to as *subjects*. *Flows* occur between a subject and a resource, and between the subject's partition and the resource's partition, in a direction defined by a *mode*. In *read* mode, the subject is the destination of the flow and in *write* mode the subject is the source of the flow (by convention, the subject is always listed first in flow statements, followed by the resource and the mode). The system enforces a partition-to-partition ($P2P_p$) policy and a subject-to-resource ($S2R_p$) policy (also known as a *least privilege* policy). Associated with each policy are the rules regarding flows, P2P and S2R respectively, which can be thought of as sets of flows:

```
flow: [s: subject, r: resource, m: mode]                    (1)
pflow: [sub_p: partition, res_p: partition, m: mode]
S2R: {flow}
P2P: {pflow}
```

3. Compound Security Policy

The SKPP was written to enforce a compound security policy, or PIFP, with requirements at the gross partition level as well as at the granularity of individual subjects and resources. The efficacy of a compound policy prompted significant discussion during the review of the SKPP. Some of this discussion follows.

3.1. Reduction of Dual Policies

It has been suggested that the dual policies can be systematically reduced offline to one statement of allowed subject-to-resource flows, by, for example, simply removing from the S2R matrix all of the flows not allowed by both policies. Certainly, this reduction would need to be performed by trusted components in order to have assurance that the correct policy is enforced. However, the clarity provided by a combination of a high level abstract policy and a granular least privilege policy would be lost. Such a reduction has been proposed in other systems as well, with similar loss. For example, one could reduce the Bell and LaPadula model's [Bell73] (1) current access set and (2) permission matrix into a simple "what is allowed" matrix for the sake of mathematical simplicity, but this would suffer the disadvantage of losing the ability to reason about the system's fundamental MAC and DAC security relationships. The maximal mathematical reduction is not always the most useful expression for engineering and comprehension.

The SKPP requires the security policy to be defined by an external configuration vector, and that this policy be bound to the evaluated product during initialization. As made clear starting in Section 1 of the SKPP, this approach to policy definition (versus, for example, a policy that is evaluated as part of the TSF) levies a responsibility on actors and elements outside of the TOE to ensure that the policy is semantically correct and has not been corrupted. While such a *configurable* mandatory security policy can be of high practical value in reducing the cost of the evaluated product, the additional product complexity, and the uncertainty as to whether a given instantiation of configuration data is well-formed, may weaken the assurance arguments. This problem is given considerable relief with the SKPP's use of a compound security policy, in which both partition-to-partition policy rules (P2P) and the subject-to-resource (S2R) policy rules apply: no matter how complex the S2R rules are, with correctly configured policy data, the information stake-holders such as the site security officer (SSO) and the accreditor have the ability to understand the compound policy by examining the P2P rules.

The $P2P_p$ policy provides the foundation for covert channel requirements. Without the $P2P_p$, the covert channel requirements, as they are written, would be vacuous.

Finally, regarding the $S2R_p$ policy, the ability of a secure system to realize the goals of accountability, as well as the confinement of damage is limited by the level of granularity with which the system is able to invoke the *principle of least privilege* [Saltzer75]. To provide high assurance, SKPP requirements for least privilege apply at the same granularity as the resources that are exported, i.e., at the resource level.

Given that an SKPP-based system is configured to enforce a compound policy, one might ask: "Are you saying that the separation kernel must literally make two separate checks before allowing any operation that invokes a flow?" The answer is: "yes," and engineering aspects are discussed in the next section.

3.2. Engineering of Compound Policies

A TOE implementation can encode the compound policies however it likes for efficiency, and cache the related decisions, as long as these encodings and caches are performed by the TSF or its initialization routines. In fact, with suitable hardware support, runtime software checks may not be needed at all, e.g., if all of the flows of each runtime application are verified during initialization to conform to the policy, and those verifications are cached in the form of hardware segment descriptors. The kernel could then pass the descriptors to applications in the form of application start-up data, and no further software checks would be required.

This approach requires that the entire compound policy must be checked by trusted mechanisms. It is not sufficient, for example, to reduce the policies in an ad hoc manner and then pass the simplified "what is allowed" matrix to the runtime separation kernel, as there would be no assurance of correctness in the reduction.

Of course, it may not be feasible to reduce access control to an initialization function, e.g., if resources are created or brought into a subject's address space during runtime, or if resources are not associated with *durable permission caches* (such as segment descriptors). Such resources might include certain encapsulated synchronization objects, for which it may be possible for the TSF to cache a "reduced" matrix representing the compound policy.

4. SKPP Policy Semantics

The SKPP compound policy semantics resulted from a long and winding road. For two years during which there was much discussion with the NSA, the *original* policy (see Equation 2) was stable. Then a revised version was submitted for review in August of 2006 (Equation 3). In September, the NSA directed the authoring team to restate the compound policy as an *ordered* system of checks with three-state access control values (viz., allow, deny, don't care), much like a DAC system (see Equation 4). By October of 2006, the *final* policy was submitted to the NSA (see Equation 5). These policies are described below. Related specifications are described in Appendix A.

4.1. *Original* Policy

The original security policy for the SKPP was fairly simple, but strict: for a flow be *ALLOWED*, it must be permitted by both the P2P policy rules and the S2R policy rules, as shown in this boolean expression [SKPPv0.621_2004_07_07]:

$$\begin{aligned} &\texttt{ALLOWED([s: subject, r:resource, m:mode])} = \\ &\quad \texttt{[s, r, m]} \in \texttt{S2R} \\ &\quad \wedge \\ &\quad \texttt{[s.p, r.p, m]} \in \texttt{P2P} \end{aligned} \tag{2}$$

This policy was revised to accommodate vendors who wished to bifurcate the compound policy [SKPP V1.0-060801-clean], allowing either or both policies to be active in a given system:

$$\begin{aligned} &\texttt{ALLOWED([s: subject, r:resource, m:mode])} = \\ &\quad \texttt{S2R}_{\texttt{p}} \in \texttt{sys.policy} \rightarrow \\ &\qquad \texttt{[s, r, m]} \in \texttt{S2R} \\ &\quad \wedge \\ &\quad \texttt{P2P}_{\texttt{p}} \in \texttt{sys.policy} \rightarrow \\ &\qquad \texttt{[s.p, r.p, m]} \in \texttt{P2P} \end{aligned} \tag{3}$$

(where sys.policy indicates which policies are configured to be active).

4.2. *Ordered* Policy

Based on a desire for the SKPP S2R policy to be consistent with DAC policies in other systems (e.g., [MLOSPP]), the SKPP management team decided that the S2R policy should be structured with a set of ordered rules by which the priority of authorizations would be determined [Badillo06]. This policy includes notions of explicit *denial* of access, and a "don't care" mode of access, such that each mode (read, write) in the policy matrix would have one of three values (allowed, denied, or null).

The ordering was as follows:

```
ALLOWED([s: subject, r:resource, m:mode]) =                                    (4)
        If S2R(s,r).m = deny
                then false
        else if S2R(s, r).m = allow
                then true
        else if P2P(s.p, r.p).m = deny  #null S2R(s, r).m
                then false
        else if P2P(s.p, r.p).m = allow
                then true
        else false                              # null P2P(s.p, r.p).m
```

This policy allows an override of the $P2P_p$ policy by including anything other than null in the S2R entry. Some of the problems with this policy that were discussed were:

- It is not the role of a protection profile to dictate how to build a system, rather its purpose is to establish requirements for what the derived system should be and do. In this case, the suggested policy appears to be telling the vendor how to implement DAC, in terms of the ordering of precedence rules.
- For the SKPP to require a specific ordering is too restrictive: it cannot possibly be the best model for every IT environment. Ideally, "the System Security Officer can decide the most suitable semantics for a specific application environment, without being forced to adopt a specific built-in policy" [Bertino99].
- The ordering seems arbitrary, as there does not appear to be a reason that the more granular permissions should have more precedence. It would seem reasonable to build a system with either S2R or P2P having precedence.
- Negative authorizations can introduce complexity (e.g., policy conflicts and potential user confusion [Barkley98]). In complex environments, monotonicity is recognized as a simplifying restriction, which ensures computability and correctness [Blaze99] as well as coherency, since a unique access decision is readily apparent [Karjoth01].

In the end, specific modes of access are not required by SKPP. It is sufficient if the means of indicating **individual** permissions allow the TSF to differentiate between authorization and denial of a requested flow. It is an exercise for the TOE developer to determine how best to achieve the intent.

4.3. *Final* Policy

The final policy [SKPP] simplified the ordered set of rules, although it is still more complicated and less restrictive than the original:

```
ALLOWED([s: subject, r: resource, m: mode]) =                                  (5)
        S2Rp ∈ sys.policy → (
                S2R(s, r). m = allow
                ∨
                 (S2R(s,r).m = null
                  ∧
                  P2P(s.p, r.p).m = allow
        )         )
        ∧
        P2Pp ∈ sys.policy →
                P2P(s.p, r.p).m = allow
```

In the $S2R_p$ policy the S2R rules override the P2P rules everywhere except where there is a null entry in the S2R rule set. Note that the $S2R_p$ policy is defined to reference the P2P values, regardless of whether the $P2P_p$ policy itself is active.

4.4. Verification and Engineering Suggestions

We believe that it is a viable option for a vendor to elect to implement the *original* policy (see Equation 2) as the default in an SKPP-based system. Since the S2R matrix cannot be ignored in the original policy, the set of secure

reachable states (or the set of accesses allowed) is a subset of those allowed by the final policy: so the original policy can be considered to be at least as secure as the final policy, insofar as it introduces no new states that would require further analysis. Of course, having implemented the original policy as a default, a vendor could also choose to allow the final policy to be selected as a configuration-vector option.

The SKPP requires the vendor to provide a *formal security policy model* of the behavior observable at the interface of the TSF. The formalization in Equation 5 precisely describes the flow, or access control, requirement of the

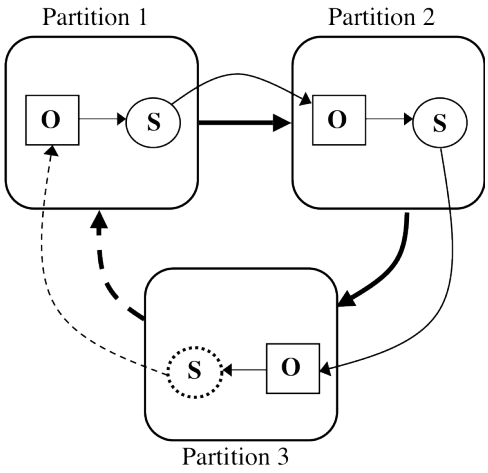

Figure 1. Cycle resolved via PAS

SKPP. A vendor may choose a different representation of the security policy in the formal model [Greve03], or add more descriptive elements to the formal model (e.g. a representation of state changes), but Equation 5 (or Equations 2 or 3) should be sufficient as the core of the policy statement in a formal security policy model of an SKPP-based system (e.g., [Levin04]).

5. Basic Acyclic Flows

Another significant concern with separation kernels is the identification of trusted subjects. Information stakeholders need assurance that all trusted subjects in deployed systems are suitably trustworthy, and that they are confined with respect to the principle of least privilege. To support the analysis of trusted subjects, the SKPP requires the identification of a *strict* base policy, or *PIFP Acyclic Subset* (PAS), an acyclic subset of the flows allowed by the P2P rules (the *PAS* terminology is introduced here to explain the SKPP requirements stated in OE.TRUSTED_FLOWS). The PAS allows the SSO to understand the core *lattice* [Denning76] of the policy, identify trusted subjects, as well as understand the scope of (*relaxed*) flows available to trusted subjects. However, if the separation kernel is designed or configured such that P2P rules are *advisory*, in the sense that they may be over-ridden by the S2R rules, then the value of the PAS as a tool will be diminished. Figure 1 shows a cycle between partitions 1, 2 and 3 (thick arrows) and between resources (thin arrows), and illustrates how a PAS (the thick, solid arrows) determines which of the subjects must be trusted (i.e., the subject in Partition 3) [Levin06].

Like the P2P matrix, the PAS can be viewed as a set of partition flows:

$$PAS = \{[subj\_p: partition, res\_p: partition, m: mode]\} \quad (6)$$

The PAS is a subset of P2P, and does not contain any cycles:

$$PAS \subseteq P2P \quad (7)$$
$$\wedge$$
$$\forall\ px: path\ (px \subseteq PAS \rightarrow \nexists\ px2: path\ (px2 \subseteq PAS \wedge (px.first = px2.last \wedge px.last = px2.first)))$$

where *path* is an ordered set (list) of partition flows in which the destination of each flow in the list is the source of each succeeding flow, thus linking the two partitions:

```
∀ p:path, i: integer (                                                    (8)
        i >= 1 ∧ i < length(p) ->
                destination(p.i) = source(p.i+1)
)
```

Since a path, here, is a total order, the set of possible paths in the PAS (i.e., the power set of paths in the PAS) may contain paths that partially overlap.

The point of PAS is to concisely identify the basic strict policy of the system, and to help ensure that the system's identified set of trusted subjects is correct. The TOE must ensure the latter property, based on the PAS (per OE.TRUSTED_FLOWS[2]).

```
∀ s: subject, r:resource, m:mode(                                         (9)
        ALLOWED([s, r, m])        # see definition of ALLOWED
        →  (
                [s.p, r.p, m] ∈ PAS  # either the flow is in PAS
                ∨
                s ∈ trusted_subjects  # or the flow is caused by trusted subject
)               )
```

5.1. Multiple Partitions per Equivalence Class

In determining the PAS, it is necessary to address the possibility of cycles among partitions. The SKPP allows two different forms of cycles among partitions:

1). A cycle that includes one or more flows that (a) are caused by a trusted subject and (b) are not in the PAS; together with zero or more flows from the PAS).

2). A cycle in which all of the flows are in the same equivalence class.

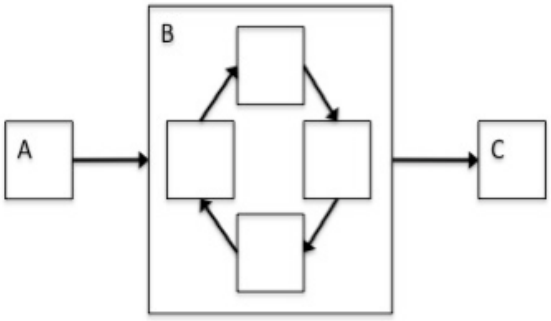

Figure 2. Acyclic Flows with Policy Equivalence Class

A TOE may be configured with multiple partitions representing a single *policy equivalence class*, where the resources in such a group of partitions would be treated equivalently with respect to the $P2P_p$ policy. For example, an MLS application running on the TOE could have multiple partitions interpreted as SECRET in the application domain. Another example of a policy equivalence class is the group of partitions labeled *B* in Figure 2; which is part of an acyclic flow from partition A to partition C. For determining whether or not the PAS is acyclic, each equivalence class included in the PAS is collapsed, so that any cycles therein are discounted.

---

[2] *… a partial order of the flows that are allowed between policy equivalence classes will be identified. Any subject allowed by the configuration data to cause information flow that is contrary to the partial order will be trusted …* [SKPP]

The abstract PAS policy is *monomorphic* to an MLS policy over the same entities, meaning that a given PAS policy can always be mapped to (a subset of) an MLS policy. Since the partition flows are acyclic, they can be embedded in a corresponding lattice (however the reverse is not necessarily true, as the lattice's transitivity characteristic might not be present). Thus, so-called *intransitive non-interference* [Rushby92] policies can be implemented with the PAS, as shown in Figure 3.

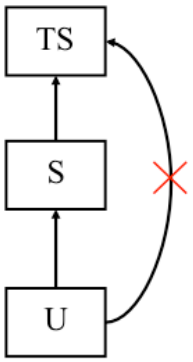

Figure 3. Intransitive Flow

5.2. Alternative Policy Model

Other models for P2P, S2R and PAS resolution are possible. For example, a compelling alternative (indicated as P2P' and S2R') is one where P2P' itself would be required to be acyclic (per Equation 7), and PAS would not be necessary. Flows allowed by S2R' rules but not by P2P' rules would be denied unless the calling subject was a trusted subject. This has the benefit of both simplifying the policy and also allowing S2R to override P2P for trusted subjects, obviating the need for a bifurcated policy (per Equation 5):

```
ALLOWED([s: subject, r:resource, m:mode]) =                    (10)
        [s, r, m] ∈ S2R'
        ∧
        ([s.p, r.p, m] ∈ P2P'  # either the flow is in P2P'
         ∨
         s ∈ trusted_subjects  # or the flow is caused by trusted subject
        )
```

6. Assurance level of SKPP

Since the publication of the SKPP, there has been some controversy regarding assurance and robustness levels of separation kernels. This section provides some historical perspective.

6.1. Robustness vs. EAL

The CC definition of assurance has changed over time. In CC Version 2.3 (against which the SKPP was evaluated), *assurance* is defined as "grounds for confidence that an entity meets its security objectives." [CCv2.3] This definition was modified in CC Version 3.1 Revision 3 (the current version) to "grounds for confidence that a TOE meets the SFRs" [CCv3.1]. The revised definition clarifies that assurance is determined through a (presumably) objective assessment of the security functionality provided by a TOE.

The CC does not address *robustness*, but the US Department of Defense IA policies define it as "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" [8500.01E, 8500.2]. The robustness of an IA solution is rated as *high, medium* or *basic* where high robustness is defined as "security services and mechanisms that provide the most stringent protection and rigorous security countermeasures" [IATF]. In other words, *robustness* is not the same as *assurance*.

For US Government protection profiles, the robustness level of a TOE is established based on the strength of the protection mechanisms implemented by the TOE to protect the data and the assurance that the design and implementation are correct. A TOE with a highest EAL rating would not automatically receive the highest robustness rating. For example, a TOE that meets EAL7 requirements but does not include hardware as part of the TOE would not be considered as a high robustness TOE. While consistency instruction manuals for development of US Government Protection Profiles have been published by the NSA for basic and medium robustness [NIAP05a, NIAP05b], no such manual exists for high robustness.

To extend the robustness or make other enhancements, a protection profile may make two types of changes to the standard (e.g., EAL6 and EAL7) assurance package requirements. The first is a*ugmentation*, where, for a given EAL, additional assurance components are drawn from the CC-defined family of assurance requirements; in contrast, the second, *extended* requirements are either modifications of existing CC requirements or completely new requirements, which are only specified when there are no equivalent requirements in the CC.

6.2. SKPP *Robustness*

The methodology used to derive the assurance requirements for the SKPP involved the following steps:

1. Assemble the initial set of requirements based on the EAL6 package.
2. Augment the base set with EAL7 requirements to attain the assurance necessary for high robustness, e.g., use of formal methods.
3. Extend the requirements to address issues not covered by the CC, e.g., platform assurance and trusted initialization.

Following steps one and two, the initial assurance requirements in the SKPP resulted in a combination of assurance components from the EAL6 and EAL7 assurance packages, which would be considered as EAL6 with augmentation (i.e., EAL6+). One example is the "depth of testing" requirement. CC Version 2.3 requires ATE_DPT.2, "low-level design" testing, for both EAL5 and EAL6, while ATE_DPT.3 is only required at EAL7. The latter adds the requirement that functional testing must be performed at the level of the implementation representation (source code). To be consistent with high robustness, the team augmented the SKPP with ATE_DPT.3.

In step three, the SKPP introduced a significant number of *extended* requirements to ensure a level of assurance that is necessary to support operating environments requiring high robustness. Some of the trusted initialization concerns raised in the SKPP were subsequently addressed in CC Version 3.1 Revision 3; in ADV_ARC.1.3C and in annex A (CCV3.1R3, Part 3). Specifically, ADV_ARC.1.3C requires the TOE developer to "describe how the TSF initialization process is secure;" and Annex A contains discussions on several ADV_INI SARs: the principles of self-protection, domain separation and non-bypassability must be applied to the initialization function (paragraph 222); the TSF must be protected against the initialization code (paragraph 533); and the initialization function must ensure the integrity to the TSF (paragraph 534).

Although early SKPP drafts discussed an EAL6+ claim for the SKPP, it was eventually realized that the SKPP should make no such claim, due to the number of extended requirements it includes and the lack of an NSA-vetted analysis indicating that those extended requirements constituted a legitimate augmentation of EAL6. Sections 1.3 and 7.9 of the SKPP explicitly state that the SKPP does not claim conformance to any EAL, which is consistent with its CC validation certificate. Specifically, the CC validation certificate identifies the SKPP as a high robustness assurance package, but does not make a claim of a specific EAL. Nevertheless, readers of the SKPP may notice that the rationale for the A.COVERT_CHANNEL assumption mentions EAL6+. This was due to an editorial oversight and should not be construed to assert an EAL6+ claim for the SKPP.

6.3. Product Evaluations

While SKPP was validated as high robustness, only, a commercial product was soon evaluated against the SKPP and received credit for achieving both high robustness, *and* "EAL6+" [VPL]. The SKPP authoring team contacted NIAP to enquire about the apparent contradiction between the SKPP assurance claim (viz., no assurance claim) and that stated in the product's CC validation certificate (EAL6+). The NIAP response [NIAP08] implied that the product Security Target (ST) constituted a legitimate mapping of High Robustness to the equivalent of EAL6 augmented because that ST "covers the intent of all EAL6 and some EAL7 requirements." This, however, begs the question about the criteria used to make such a determination, since the evidence presented in the product's ST and in its ST Evaluation Report made no EAL claims, and there has been no statement by the CC community that describes how extended requirements translate to an EAL.

7. Conclusions

To help future developers understand the meaning and intent of the SKPP, we have provided a historical perspective and various rationale for a set of key issues regarding the SKPP. A preliminary set of errata with respect to the published version of SKPP is provided in Appendix B, and we hope to provide a more thorough examination in the near future.

References

8500.01E Department of Defense Directive Number 8500.01E, October 24, 2002, Certified Current as of April 23, 2007.

8500.2 Department of Defense Instruction Number 8500.2, February 6, 2003.

Badillo06 Badillo, H, "Summary of Proposed Change to the PIFP," e-mail communication with authoring team, September 29, 2006.

Barkley98 J. Barkley and A. Cincotta, "Managing role/permission relationships using object access types." *In Proceedings of the Third ACM Workshop on Role-Based Access Control*, pp. 73-80, 1998.

Bell73 D. E. Bell and L. LaPadula, "Secure computer systems: Mathematical foundations and model," Tech. Rep. M74-244, The MITRE Corp., Bedford, MA, May 1973.

Bertino99 E. Bertino, F. Buccafurri, E. Ferrari, and P. Rullo, A logical framework for reasoning on data access control policies, *In. Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pp. 175 – 189, June 1999.

Blaze99 M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis, "The Role of Trust Management in Distributed Systems Security," Lecture Notes in Computer Science, Issue 1603, pp. 185-210, 1999.

CCv2.3 Common Criteria for Information Technology Security Evaluation, Version 2.3, CCIMB-2005-08-[001, 002, 003], August 2005.

CCv3.1 Common Criteria for Information Technology Security Evaluation, Version 3.1 Revision 3, CCIMB-2009-07-[001, 002, 003], July 2009.

Denning76 D. E. Denning, "A Lattice Model of Secure Information Flow," *Communications of the A.C.M.*, vol. 19, no. 5, pp. 236–243, 1976.

Greve03 D. Greve, M. Wilding, and W. M. Vanfleet, "A separation kernel formal security policy," in *Fourth International Workshop on the ACL2 Theorem Prover and Its Applications*, (Boulder, Colorado), July 2003.

IATF Information Assurance Technical Framework, Chapter 4, Release 3.1, National Security Agency, September 2002.

Karjoth01 G. Karjoth, "The Authorization Service of Tivoli Policy Director," *Proceedings of the 17th Annual Computer Security Applications Conference,* pp. 319- 328, December 2001.

Levin 06 T. Levin, C. Irvine and T. Nguyen, "An Analysis of Three Kernel-based Multilevel Security Architectures", NPS Technical Report NPS-CS-06-001, August 2006.

Levin04 T. E. Levin, C. E. Irvine, and T. D. Nguyen, "A Least Privilege Model for Static Separation Kernels", NPS-CS-05-003, Naval Postgraduate School, October 2004.

MLOSPP Information Assurance Directorate, National Security Agency, Fort George G. Meade, MD 20755-6000, U.S. Government Protection Profile for Multilevel Operating Systems in Medium Robustness Environments, Version 1.91, Mar. 2007.

NIAP08 Email correspondence between NIAP / CCEVS representative and authoring team, December 11, 2008.

NIAP05a Consistency Instruction Manual for Development of U.S. Government Protection Profiles for use in Basic Robustness Environments, National Security Agency Information Assurance Directorate, Release 3.0, Fort Meade, MD, 1 February 2005. http://www.niap-ccevs.org/pp/basic_rob_manual-3.0.pdf

NIAP05b Consistency Instruction Manual for Development of U.S. Government Protection Profiles for use in Medium Robustness Environments, National Security Agency Information Assurance Directorate, Release 3.0, Fort Meade, MD, 1 February 2005. http://www.niap-ccevs.org/pp/med_rob_manual-3.0.pdf

PKPP The Open Group, Protection Profile for Partitioning Kernels in Environments Requiring Augmented High Robustness. Not Published, 1.3 ed., 2003.

Rushby92 J. Rushby, "Noninterference, transitivity, and channel control security policies," Tech. Rep. CSL-92-02, Computer Science Laboratory, SRI International, Menlo Park, CA, December 1992.

Saltzer75 J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

SKPP U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness. No. Version 1.03, National Security Agency, June 2007.

VPL NIAP, "Validated Products List," accessed 10/10/10: http://www.niap-ccevs.org/vpl.

Appendix A: SKPP Versions

Several versions of the SKPP are referenced in the text to provide readers with a notion of the development timeline. The review drafts were publically distributed to a broad audience. Although not publically accessible to our knowledge, the sponsor may choose to make them available.

SKPPv0.621_2004_07_07 – first review, original policy

SKPP V1.0-060801-clean – second review, bifurcated policy

SKPP, June, 2007 – final version, final policy

Appendix B: Errata

This section summarizes several suggested corrections to the SKPP, intended as input for a future revision.

EAL

A.COVERT_CHANNEL mentions EAL6+. This was an editorial oversight and should be removed so as not be construed to assert an EAL6+ claim for the SKPP.

External vs. Exported

Figure 2-7 contains a typographical error occurs in. The term e*xternal* should read, *exported*, instead. "Resources" include internal resources that are reserved by the kernel for its own use and resources that the kernel exports. There are no "external" resources.

Acyclic vs. Partial Order

In the SKPP, the "PAS" concept is described as a *partial order*. While a partial order is sufficient, it is not necessary for the identification of the *strict* policy. The PAS should be referred to as *acyclic* because, while a partial order allows no circular flows between partitions, it requires an explicit relation for all transitive flows; however, it may not be desirable for an SKPP system policy to include all of the transitive relationships (flows) that its basic flows imply.

Suggested modifications to A.TRUSTED_FLOWS and OE.TRUSTED_FLOWS follow, with deletions in *striked* font and additions in bold font.

> A.TRUSTED_FLOWS. For any subject configured to have the **ability to cause a partition flow that is not included in the *PIFP Acyclic Subset*** ~~unrestricted access in multiple policy equivalence classes~~, it is assumed that the subject is trusted at least with assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.
>
> OE.TRUSTED_FLOWS. For each configuration of the TOE, a ***PIFP Acyclic Subset*** ~~partial order~~ **will be identified, which is a subset** of the flows that are allowed between policy equivalence classes, **and which is acyclic** ~~will be identified~~. Any subject allowed by the configuration data to cause information flow that is contrary to the ***PIFP Acyclic Subset*** will be trusted at least with

assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.

Inconsistencies from Policy Change

Areas of the SKPP that are inconsistent with respect to the *final* policy are:

a. Rationale Section, e.g. AVA_CCA_EXP.2 assumes $P2P_p$ and $S2R_p$ are equally enforced

b. Error from Section 2: *The least privilege abstraction requires that both partition-pair and subject-exported resource pair authorizations are used to determine if a flow mode is allowed.*

   Whereas, in fact, either policy can be left out through configuration choice.