Separation Kernel Protection Profile Revisited: *Choices and Rationale*

Layered Assurance Workshop 2010

Austin, Texas

Timothy Levin, Thuy Nguyen,
Cynthia Irvine, Michael McEvilley

Overview

- Purpose
  - Help users of SKPP understand intent of requirements
  - Explain critical decisions in SKPP development
  - Describe several errata
- Content
  - SKPP Accomplishments
  - Separation Kernels
  - Compound Security Policy
  - Evolution of Security Policy Semantics
  - Acyclic Partition Flow Requirement
  - SKPP-based Assurance Level
  - SKPP Errata

SKPP Accomplishments

- First high robustness product protection profile sanctioned by the U.S. Government.
- Specified new requirements for Target of Evaluation (TOE) hardware.
- Developed a new abstraction and related requirements for least privilege in separation kernels
- Developed a means by which a TOE could ensure the adequacy of its trusted subjects
- Developed configuration vector concepts
  - Multiple vectors allow pre-vetted policy options
- Developed specifications for a configuration tool
  - used by security administrators to prepare configuration vectors .
- Developed concepts and requirements for dynamic, runtime changes to the TOE configuration
- Finished

Separation Kernel

- Controls all physical resources
- Exports subset of resources
- Kernel partitions exported resources
  - Every resource bound to exactly one partition
- Compound Policy: $S2R_p$ and $P2P_p$
  - Controls flow between subjects and resources
    - Flow = [s: subject, r: resource, m: mode]
    - Allowed flows: S2R {flow}   // Policy structure
  - Controls flows between partitions
    - Pflow = [subj_p: partition, res_p: partition, m: mode]
    - Allowed flows: P2P {pflow}   // Policy structure

Compound Security Policy: Engineering Choices

- Reduce dual policies with **<u>trusted</u>** initialization function:
  - allowed (s: subject, r: resource m: mode) means that
    - flow(s, r, m) is allowed by S2R and P2P // details later
  - Cache all legal accesses in hardware during initialization:

```
∀ s: subject, r: resource m: mode:
    allowed (s, r, m) ->
      cache(s,r,m) = valid
      export (s, cache(s,r,m))
```

  - Cached accesses checked by hardware during runtime
    - subject s reads resource r using hardware token, cache(s,r,m)
  - No need for kernel to access P2P or S2R during runtime
    - Except for encapsulated objects w.o. hw descriptors

SKPP Policy Semantics: Original Policy

- Original Policy

  ALLOWED([s: subject, r:resource, m:mode]) =

  $[s, r, m] \in S2R$

  $\wedge$

  $[s.p, r.p, m] \in P2P$

- Bipartite Policy

  ALLOWED([s: subject, r:resource, m:mode]) =

  $S2R_p \in sys.policy \rightarrow$

  $[s, r, m] \in S2R$

  $\wedge$

  $P2P_p \in sys.policy \rightarrow$

  $[s.p, r.p, m] \in P2P$

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

SKPP Policy Semantics: Ordered Policy

- Ordered Policy
  - Includes three-value logic: *allow*, *deny* and *don't care* (null)

```
ALLOWED?([s: subject, r:resource, m:mode]) =
    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 enables an override of the P2P$_p$ policy by including anything other than null in the S2R entry.

SKPP Policy Semantics: Final Policy

- Final (Published) Policy

$S2R_p \in sys.policy \rightarrow ($

$\quad S2R(s, r).\ m = allow$

$\quad \vee$

$\quad (S2R(s,r).m = null$

$\quad\quad \wedge$

$\quad P2P(s.p, r.p).m = allow\ )\ )$

$\wedge$

$P2P_p \in sys.policy \rightarrow$

$\quad P2P(s.p, r.p).m = allow$

- Note that the $S2R_p$ policy references the P2P values, regardless of whether the $P2P_p$ policy itself is active.

SKPP Policy Semantics: Engineering Choices

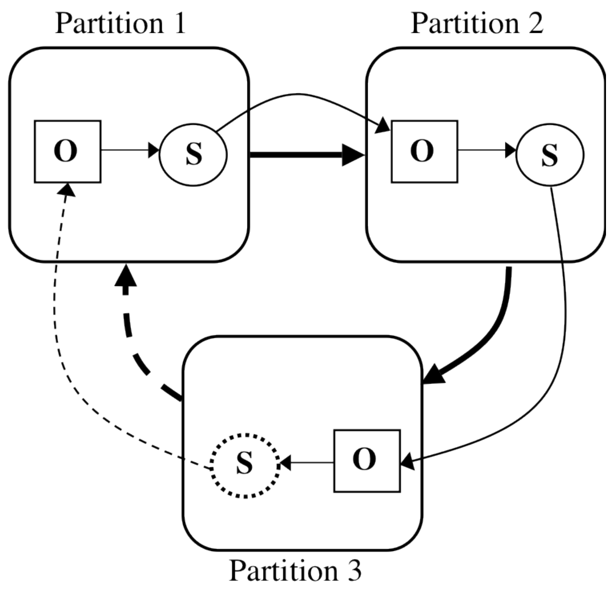
- Formal security policy model required
  - Vendor can select from variety of noninterference and other models
  - NSA seems to have encouraged GWV model
- Policy implementation for SKPP compliance
  - Implement *original* policy or *final* policy?
    - Evaluators may require final policy be available as a configuration option
    - Original policy is at least as restrictive
      - Lack of policy override means that there are fewer reachable states in the original policy than in the final policy

Acyclic Partition Flow Requirement

- Environment Security Objectives
  - OE.TRUSTED_FLOWS
    - *For each configuration of the TOE, 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 at least with assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.*

Acyclic Partition Flow Requirement

- OE.TRUSTED_FLOWS requires that the TSF be presented with a representation of the *strict* policy of the system
  - Subset of flows allowed by P2P rules
  - Only *trusted subjects* may bypass strict policy
  - Legend
    - Strict P2P Policy ———
    - Trusted Subject ◌
    - Relaxed flows - - -



Alternative Policy Model

- Other models for P2P, S2R and PAS resolution are possible.
- One is where P2P (P2P') would be required to be acyclic
  - Flows allowed by S2R but not by P2P' would be denied
    - unless the calling subject was a trusted subject.
- This simplifies the policy and also allows S2R to override P2P for trusted subjects

```
ALLOWED([s: subject, r:resource, m:mode]) =
        [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
        )
```

Assurance level of SKPP

- To extend robustness a protection profile may use:
  - *Augmentation*
    - Unmodified assurance components drawn from the CC-defined family of assurance requirements
  - *Extended requirements*
    - modifications of existing CC requirements or
    - completely new requirements
- SKPP used both
- SKPP made no EAL claim
  - Many extended requirements
    - E.g., ATE_DPT.3, ADV_ARC.1.3C
  - Lack of an standard for how extended requirements equate to augmentation of EAL6

Eratta: Miscellaneous

- **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 *external* should read, *exported*, instead.
    - "Resources" include internal resources and resources that the kernel exports. There are no "external" resources.

Eratta: Acyclic vs. Partital Order

- In the SKPP, the acyclic concept is described as a *partial order*.
  - partial order is sufficient, but too strong.
  - Change SKPP to use *acyclic*
    - not be desirable to require SKPP systems to include all of the transitive relationships (flows) that its basic flows imply.

Eratta: Inconsistencies from Policy Change

- Areas of the SKPP that are inconsistent with respect to the *final* policy are:
  - Rationale Section, e.g. AVA_CCA_EXP.2 assumes $P2P_p$ and $S2R_p$ are equally enforced
  - 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.

Separation Kernel Protection Profile Revisited: Choices and Rationale

Timothy E. Levin, levin@nps.edu
Thuy D. Nguyen, tdnguyen@nps.edu
Cynthia E. Irvine, PhD, irvine@nps.edu
Michael McEvilley, mcevilley@mitre.org

Center for Information Systems Security Studies and Research
Department of Computer Science
Naval Postgraduate School, Monterey, CA 93943
U.S.A

http://cisr.nps.edu