On the Formal Definition of Separation-of-Duty (SoD) Policies and their Composition

Virgil D. Gligor
Department of Electrical Engineering
University of Maryland
College Park, Maryland 20742
gligor@eng.umd.edu

Serban I. Gavrila
VDG Inc.
6009 Brookside Drive
Chevy Chase, Maryland 20815
gavrila@csmes.ncsl.nist.gov

David Ferraiolo
NIST
US Department of Commerce
Gaithersburg, Maryland 20899
ferraiolo@csmes.ncsl.nist.gov

May 5, 1998
SoD premise:

• Violations that require collusion are less likely to happen

SoD goals:

• Separate sensitive tasks of an application such that integrity violations => collusion
• Minimize risk of collusion
  by *careful* assignment of users to separate tasks

SoD implementation:

• Define integrity property of an application
• Partition application into separate operations and objects
• *carefully* assign of users to separate application partitions
SoD Policies

Advantage:

- wide-spread acceptance by business, industry, government

Drawbacks:

- *application-oriented policy*
  => limited scope
  => separate administration

- *family of policies*
  => required system flexibility

- *uncertain policy interpretation*
  => uncertain relative strength
Drawbacks:
- application-oriented policy
  => limited scope
  => separate administration
- family of policies
  => required system flexibility
- uncertain policy
  interpretation
  => uncertain relative strength

Mitigation:
- make it a feature of a global policy
- provide administrative tools
- define formally

Solution: Define, implement, and administer SoD policies in systems supporting Role-Based Access Control (RBAC)

\{ users \} \rightarrow \{ roles \} : \{ operations \} \rightarrow \{ objects \}
Vision: SoD Administrative Tool

GUIs

A P I s

SoD Policy Library

SoD Policy 1

SoD Policy n

Oracle RBAC Semantics

RBAC/DTOS Semantics

Other RBAC Semantics

RBAC/Web Semantics

Mapper to Oracle’s SQL

Mapper to RBAC/Synergy commands

Mapper to Other RBAC APIs

Mapper to RBAC/Web APIs

Oracle RBAC Server

SSCL

RBAC/DTOS security module

APIs

Perl

Other RBAC Server

RBAC/Web scripts

Administrative Tool

SoD Policy Server

Oracle RBAC

SQL
Systems

- **state machine**
  STATES, SUBJECTS, USERS, OPERATIONS, OBJECTS

- **state transitions**
  - commands: $op(s_1, S, obj, s_2)$
  - command sequence: $op_1(s_0, S_1, obj, s_1) op_2(s_1, S_2, obj, s_2) ...$
  - tranquil commands: do not alter security attributes

- **system**: a set of command sequences with start states $s_0$ in STATES$_0$.

- **secure state, commands**: those that satisfy properties

- **reachable state**: a state appearing in a command sequence of a system

- **secure system**: all state transitions and reachable states are secure

- $\Omega$: set of all command sequences of a secure system
Applications and Executability

• application: \( \text{App} = [\text{ObjSet}, \text{OpSet}, \text{Plan}] \)

  - \textbf{plan: a finite set of pairs} \( \{(\text{obj}_i, \text{op}_i)\} \)
  - ordered plan: an ordered set of pairs \( \{(\text{obj}_i, \text{op}_i)\} \)
  - plans with “operation bracketing” (e.g., least-privilege princ.)

  \[ \ldots \]

• \( \text{App}_1 \cup \text{App}_2 = \)
  \[ [\text{ObjSet}_1 \cup \text{ObjSet}_2, \text{OpSet}_1 \cup \text{OpSet}_2, \text{Plan}_1 \cup \text{Plan}_2] \]

• command sequence \( \sigma \) \textit{executes} \( \text{App} \) if for any pair \( (\text{obj}_i, \text{op}_i) \) in Plan there is a command \( \text{op}_i(s_k, S, \text{obj}_i, s_{k+1}) \) in \( \sigma \)
Property Types

\[ P = \text{Attribute (AT) properties} \wedge \]

\[ \text{Access Management (AM) properties} \wedge \]

\[ \text{Access Authorization (AA) properties} \]
Examples of Property Types

• **Attribute (AT) Properties**
  – security (integrity) levels, partial order, lattice property
  – roles, hierarchy, permissions, membership, inheritance

• **Access Management (AM) Properties**
  – distribution, review, revocation of permissions
    • selectivity, transitivity, independence ...
  – object / subject creation and destruction
  – object encapsulation

• **Access Authorization (AA) Properties**
  – required subject and object attributes for access
    • BLP, Biba, RBAC, UNIX ...
Property Dependencies

“uses”

other types of dependencies exist

Individual policy properties cannot be composed independently
Policy Structure

\[ P = P \land \text{Admin}(P) \land \text{Compat}(P, \text{App}) \]

- Access management
- Access authorization
- Attribute properties

Safety Properties

Safety or Liveness Properties ?
SoD Policy Structure

SoD-P = SoD-P ∧ Admin(SoD-P) ∧ Compat(SoD-P, App) ∧ RBAC-P
Admin(P)

P: a set of tranquil command sequences with the start state in $\text{STATES}_0$

for all

$Admin(P) =$ “for each $s$ in $\text{STATES}$, there exists $s_0 \in \text{STATES}_0$, there exists $\omega \in \Omega$ such that: $\omega$ starts in $s$, and $\omega$ reaches $s_0$ and $s_0^*$ is in $P$”

Compat(P, App)

$Compat(P) =$ “there exists $s_0 \in \text{STATES}_0$ and $\sigma \in P$ starting in $s_0$ such that $\sigma$ executes $\text{App}$”

.... neither Safety nor Liveness ....
Mandated Compatibility
Types of Compatibility

- **Totally multi-path Compatible**
- **Multi-path Compatible**
- **Machine Closed Compatible**
- **Strongly Compatible**

Safety-Liveness Framework

\[ \text{Compat}(P, \text{App}) \]
**Totally Multi-path Compatible**
For each start state \(s_0\) there is a command sequence \(\sigma\) in \(P\) starting in \(s_0\), and for each finite command sequence \(\sigma\) in \(P\) there is a command sequence \(\tau\) such that \(\sigma\tau\) is in \(P\) and executes \(App\).

**Machine-Closed Compatible**
For each finite command sequence \(\sigma\) in \(P\) there is a command sequence \(\tau\) such that \(\sigma\tau\) is in \(P\) and executes \(App\).

**Multi-path Compatible**
There is a start state \(s_0\) such that for each finite command sequence \(\sigma\) in \(P\) starting in \(s_0\) there is \(\tau\) such that \(\sigma\tau\) is in \(P\) and executes \(App\).

**Totally Compatible**
For each start state \(s_0\) there is a command sequence \(\sigma\) in \(P\) starting in \(s_0\) such that \(\sigma\) executes \(App\).

**Strongly Compatible**
For each start state \(s_0\) such that \(s_0^*\) is in \(P\), there is a command sequence \(\sigma\) in \(P\) starting in \(s_0\) that executes \(App\).

**Compatible**
There is a start state \(s_0\) and a command sequence \(\sigma\) in \(P\) starting in \(s_0\) that executes \(App\).
Types of Compatibility

- **Totally multi-path Compatible**
- **Multi-path Compatible**
- **Machine-Closed Compatible**
- **Overly Restrictive $STATES_0$**
- **Overly Restrictive $\sigma_s$**

Compat($P$, App)

May Require Administrative Work for App’s Execution in $P$

Gligor, Gavrila, Ferraiolo - 5/5/98
Overly Restrictive $\sigma_s$

Example:

$App = [ \{obj\}, \{op_1, op_2\}, plan ]; plan = \{(obj, op_1), (obj, op_2)\}$

$P: \text{“}u_1 \text{ and } u_2 \text{ are the only users who may execute } App \text{ and a user may not execute two distinct (or all) operations on the same object“}$

Compat($P, App$) is true

$$
\sigma = \begin{array}{c}
S_0 \\
S_1 \\
S_2
\end{array}
\begin{array}{c}
\leftarrow \\
\rightarrow \\
\rightarrow
\end{array}
\begin{array}{c}
S_1:(op_1, obj) \\
S_2:(op_2, obj)
\end{array}
$$

$u_1: (op_1: obj)$, $S_1 = \text{subject}$

$u_2: (op_1, op_2: obj)$, $S_2, S_2' = \text{subjects}$

Compat$_M(P, App)$ is false

$$
\sigma' = \begin{array}{c}
S_0 \\
S_1 \\
S_2
\end{array}
\begin{array}{c}
\leftarrow \\
\rightarrow \times \\
\rightarrow \times
\end{array}
\begin{array}{c}
S_1:(op_1, obj) \\
S_2:(op_2, obj)
\end{array}
$$

$u_1: (op_1: obj)$, $S_1$

$u_2: (op_1, op_2: obj)$, $S_2, S_2'$

$S_1:(op_2, obj)$
Simple Policy Composition

\[
P_1 = P_1 \land \text{Admin}(P_1) \land \text{Compat}(P_1, \text{App}_1)
\]
\[
P_2 = P_2 \land \text{Admin}(P_2) \land \text{Compat}(P_2, \text{App}_2)
\]

Let \( CS(P_i) = P_i \), if \( \text{Admin}(P_i) \land \text{Compat}(P_i, \text{App}_i) \) is True;
\( \varnothing \), otherwise.

(Emerging policy) \( P_1 \circ P_2 = \)
\[
= P_1 \land P_2 \land \text{Admin}(P_1 \land P_2) \land \text{Compat}(P_1 \land P_2, \text{App}_1 \cup \text{App}_2)
\]

\( P_1, P_2 \) are composable if and only if
\( CS(P_1 \circ P_2) \neq \varnothing \) whenever \( CS(P_1), CS(P_2) \neq \varnothing \)
SoD Properties (1)

Static SoD

Dynamic SoD

role membership

Strict Static SoD

role activation

1-step Strict Static SoD

Gligor, Gavrila, Ferraiolo - 5/5/98
SoD Properties (2)

Operational Static SoD

\[ \text{OpSet} = \{\text{opi}, \text{op}_j, \text{op}_k\} \]

Operational Dynamic SoD

per-Role Operational Static SoD

role activation
SoD Properties (3)

Object-based Static SoD

Object-based Dynamic SoD

per-Role, Object-based Static SoD

object access
SoD Properties (4)

History-based Dynamic SoD

\[ \text{OpSet} = \{ \text{op}_i, \text{op}_j, \text{op}_k, \text{op}_l \} \]
Relationships among SoD Properties

HDSoD

ObjDSoD  and  DSoD  and  ROpSSoD

ObjSSoD  and  SSoD  and  OpSSoD

1sSSSoD
Example: Non-Composable Separation-of-Duty Policies

Static SoD

R₂

read
write
sign

verify

R₁

read
write

R₃

sign

Operational Static SoD

Purchasing Staff
Department

Purchasing Staff
Central Administration

Static SoD

R₂

read
write
sign

verify

R₁

read
write

R₁'

read
write

R₃

sign

Operational Static SoD

Policy-Management Change