Bell-La Padula: Modelo de Confidencialidad

Cristian Adrián Ontivero
2018-05-06

Durante los comienzos de los 70, la Fuerza Aérea de Estados Unidos tenía sus preocupaciones con respecto a la seguridad de los sistemas de computadoras centrales de tiempo compartido (*time-sharing mainframe systems*). En respuesta a esto, David E. Bell y Leonard J. La Padula propusieron en una serie de reportes un modelo de política de seguridad [1, 2, 3]. Lo desarrollado en estos fue posteriormente resumido y adaptado a Multics en un reporte final [4]. El modelo Bell-La Padula (BLP) es una política de seguridad multinivel (*multilevel security policy*) basado en máquinas de estado. Define formalmente qué significa que un estado sea "seguro", y analiza qué transiciones de estado son permisibles de forma que un estado seguro no lleve a uno inseguro. El objetivo de BLP es prevenir la divulgación indebida de información; en una palabra, confidencialidad. El problema esencial que busca resolver es controlar el acceso de entidades activas llamadas sujetos, a un conjunto de entidades pasivas (protegidas) llamadas objetos. Más formalmente, el modelo consta de:

- Un conjunto de sujetos S (*subjects*).
- Un conjunto de objetos O (*objects*).
- Una matriz de accesos (*access matrix*), donde la celda m_{ij} contiene los permisos de acceso del sujeto s_i sobre el objeto o_j.

	o_1	$\cdots$	o_m
s_1	r		-
$\vdots$		$\ddots$	
s_n	rw		e

Tabla 1: Ejemplo de matriz de acceso.

  A los distintos permisos se los conoce como atributos o modos de acceso, y pueden ser e, r, a, o w.
- Un conjunto de clasificaciones de seguridad, con un orden parcial $\leq$. Las clasificaciones son, en orden creciente: *Unclassified* (U), *Confidential* (C), *Secret* (S) y *Top Secret* (TS).
- Un conjunto de categorías (*categories*).

Los sujetos suelen ser usuarios, o más generalmente procesos. Los objetos son habitualmente archivos, dispositivos y otras entidades implementadas por un sistema operativo. El conjunto de atributos de acceso surge de que hay dos efectos que un acceso puede tener sobre un objeto:

1. Extraer información ("observar" el objeto).
2. Introducir información ("alterar" el objeto).

Por lo tanto, hay cuatro tipos generales de acceso imaginables:

1. Ni observar, ni alterar; acceso e.
2. Observar sin alterar; acceso r.
3. Alterar sin observar; acceso a.
4. Observar y alterar; acceso w.

Como es deducible, las letras usadas para los atributos de acceso derivan de `execute`, `read`, `append` y `write`, pero su significado no necesariamente concuerda con el mencionado anteriormente. Por ejemplo, ejecutar un programa requiere que la computadora lea (observe) las instrucciones del mismo, y de hecho en Multics el atributo e requiere tanto permisos de ejecución como lectura. Distintas implementaciones modifican o agregan permisos, pero estos cuatro son los básicos.

La clasificación (a veces llamado *sensitivity level*), busca separar la información de acuerdo a qué tan sensible es (en un ámbito militar, en base al costo posible de que la información se filtre a un enemigo). Si bien los cuatro niveles listados son los originales del modelo, se puede usar cualquier cantidad de niveles. Por su parte, las categorías surgen del principio *need-to-know*. La idea es que no debería encomendarsele información clasificada a alguien salvo que además de tener la autorización suficiente, tenga alguna necesidad particular de conocerla relacionada a su trabajo. Por ejemplo, el teniente general del Ejercito Argentino, y el almirante de la Armada Argentina pueden ambos tener la misma autorización de seguridad (e.g. *Top Secret*), pero uno no tiene (en general) porqué tener acceso a la información del otro. Para contemplar esto se agregan las etiquetas denominadas categorías. A cada sujeto y objeto se le asigna un nivel de seguridad (*security level*), que es un par (Clasificación, Categorías). Para que un sujeto pueda observar un objeto, es necesario que su nivel de seguridad domine al nivel de seguridad del objeto. La relación de dominancia $\mathsf{dom}\colon L \times L \to \{0,1\}$ es un orden parcial, y se define como:

$$(c_1,\ K_1)\ \mathsf{dom}\ (c_2,\ K_2) \iff c_1 \geq c_2\ \wedge\ K_1 \supseteq K_2$$

donde c_1, c_2 son clasificaciones, y K_1, K_2 son conjuntos de categorías. El conjunto partes $\mathcal{P}(K)$ de las categorías junto con la relación $\subseteq$ forman un retículo. Por ejemplo, dado el conjunto $K = \{\text{A, B, C}\}$ de categorías, se tiene el retículo de la Figura 1, donde $\emptyset \subseteq \{\text{A}\}$, $\{\text{A}\} \subseteq \{\text{A,B}\}$, pero $\{\text{A}\} \not\subseteq \{\text{B}\}$, etc.; el supremo es {A, B, C}, y el ínfimo es $\emptyset$. También forman un retículo los niveles de seguridad (pares clasificación-categorías) con la relación de dominancia dom.

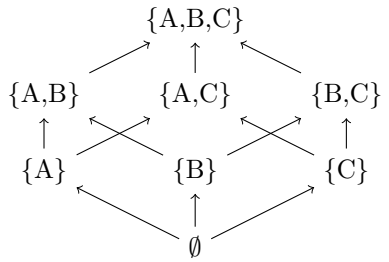

Figura 1: Diagrama de Hasse del retículo formado por $\mathcal{P}(K)$ y la relación de orden $\subseteq$, representada mediante flechas.

Haciendo abuso de notación[1] por simplicidad, denotamos con $L(s)$ el nivel de seguridad de un sujeto s, y con $L(o)$ el nivel de seguridad de un objeto o. Acordado esto, se definen dos características del sistema, y se refiere colectivamente a estas como "seguridad". La primera de estas es:

1. **Propiedad simple de seguridad[2]:** s puede leer o sii $L(s)$ dom $L(o)$ y s tiene permisos para leer o.

   Es decir, ningún proceso puede leer datos en un nivel mayor. En la primera versión del modelo ([1]), seguridad se refería solo a esta propiedad. Pero, ¿qué pasa si por ejemplo un usuario se encuentra infectado con un troyano, y accede a un documento? Este programa podría escribir en un documento a un nivel de seguridad menor que el documento original, efectivamente filtrando información como se observa en la Figura 2. Por esto, se agregó luego la propiedad siguiente.

2. **Propiedad-* (estrella)[3]:** s puede escribir o sii $L(o)$ dom $L(s)$ y s tiene permisos para escribir o.

   Es decir, ningún proceso puede escribir datos en un nivel menor.

[1] Ya que los dominios son distintos. Formalmente, se hace uso de dos funciones $f_O\colon O \to L$ y $f_S\colon S \to L$, donde L es el conjunto de niveles de seguridad.

[2] *Simple security property (ss-property)*, también conocida como *no-read-up policy* (NRU).

[3] *Star property (*-property)*, también conocida como *no-write-down policy* (NWD). Sobre el origen del nombre, Bell dijo: *I scribbled the heading "*-property" on the blackboard [..] After a burst of energetic discussion, I pointed out that if we didn't change the name right then, we'd be stuck with it forever. Nothing came to us and we continued our discussion. "*-property" it remained.* [5]

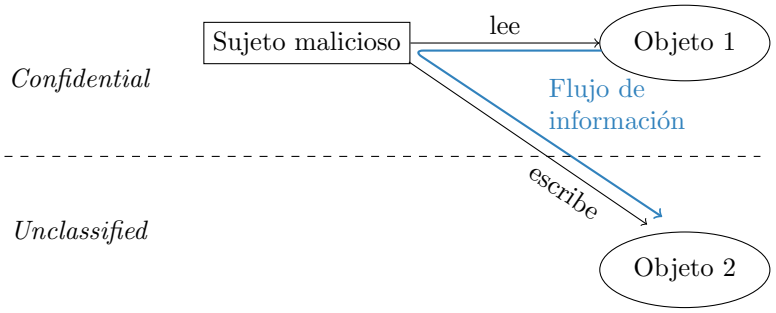

Figura 2: Flujo indebido de información de una clasificación de mayor nivel a una de menor. Este escenario motiva la propiedad-*.

Ambas propiedades combinan acceso mandatorio (e.g. "$L(o)$ dom $L(s)$") con acceso discrecional (e.g. "s tiene permisos para leer o"). El modelo como resumido por Bell y La Padula en [4] evita combinar ambas, haciendo uso de una propiedad más que llaman propiedad de seguridad discrecional (*discretionary security property*, abreviada *ds-property*).
Que un sujeto tenga permiso para leer un objeto implica un flujo de objeto a sujeto, por lo que $L(s)$ dom $L(o)$ también se puede expresar como $L(o) \rightarrow L(s)$ (es decir, $L(o)$ puede fluir a $L(s)$). Análogamente, permiso de escritura implica un flujo de sujeto a objeto, siendo así el requerimiento $L(o)$ dom $L(s)$ expresable como $L(s) \rightarrow L(o)$ [6]. Efectivamente, en BLP la información sólo puede fluir hacia arriba, no hacia abajo (salvo que una persona autorizada deliberadamente decida desclasificarla). Cabe destacar que los accesos discrecionales restringen a los mandatorios, pero no pueden contradecirlos (es decir, el mandatorio tiene prioridad sobre el discrecional).
La razón por la que se suele confiar en la seguridad de sistemas basados en este modelo, es el siguiente teorema:

Teorema Básico de Seguridad

Sea Σ un sistema con un estado inicial seguro δ_0, y T un conjunto de transformaciones de estado. Si todo elemento de T preserva la propiedad-ss y la propiedad-*, entonces $\forall i \geq 0, \delta_i$ es seguro.

Obviaremos la demostración, pero quien estuviera interesado puede verla en los reportes originales.

Ejercicio 1. Dado un modelo BLP con:

- Los siguientes sujetos, cada uno con su respectivo nivel de seguridad:

Sujeto	Nivel de Seguridad
Presidente	(TS, {N, E})
Coronel	(S, {N, E})
Mayor	(C, {E})
Soldado	(U, {N})

- Un conjunto de categorías $C = \{\text{N}, \text{E}\}$ (por nuclear, y ejercito).
- Los siguientes objetos con sus clasificaciones y las categorías como se pueden inferir de sus nombres:

Objeto	Clasificación
Código nuclear	TS
Posición del ejercito	S
Cantidad de soldados	C
Cantidad de unidades nucleares	C
Costo del programa nuclear	U
Costo del ejercito	U

- Y asumiendo que los sujetos tienen los permisos discrecionales pertinentes.

,

1. Dibujar un diagrama de Hasse de los niveles de seguridad.
2. ¿Puede el Presidente calcular el costo total de defensa (nuclear + ejercito)?
3. ¿Puede el Mayor calcular el número total de unidades nucleares y del ejercito (soldados)?
4. ¿Y el coronel?
5. ¿Puede el coronel cambiar la posición del ejercito?
6. ¿Puede el mayor cambiar el código nuclear? ¿Y el soldado?
7. ¿A qué problema hace referencia la pregunta anterior?

Propiedad de Tranquilidad

En 1987 McLean causó un debate al ejemplificar un sistema con una transición de estado que bajaba el nivel de seguridad a todos los sujetos y objetos al menor nivel, y llenaba la matriz de control de accesos con todos los permisos en cada entrada [7]. Bajo las definiciones de BLP, el estado alcanzado por el modelo es seguro. Hay dos opiniones sobre si es o no apropiado considerar semejante estado como seguro:

1. En contra de BLP (McLean): Intuitivamente, si a un sistema se lo puede llevar a un estado en el que todos pueden leer todo, no es seguro.
2. A favor de BLP (Bell): Si los requerimientos del usuario requieren tal transición de estado, entonces debería permitirsela. De lo contrario, no debería implementarsela.

Si se permite modificar los niveles de seguridad de sujetos y objetos, se puede efectivamente violar la seguridad:

↑ Un espía podría hacer copias de los objetos a los que tiene acceso. Si posteriormente se decide aumentar el nivel de seguridad de uno de estos, de modo tal que el espía ya no tenga acceso al mismo, el espía puede tomar esto como indicio de que el archivo tiene información sensible y filtrar su copia, efectivamente violando el principio-ss.

↓ Reducir el nivel de seguridad de un objeto esencialmente viola el principio-*. Para esto se suele tener sujetos de confianza (*trusted subjects*), a quienes no aplica el principio-* y están encargados de la sanitización de la información. Cómo proceder a la hora de reducir el nivel de un objeto (hacer *down-grade*) se conoce como el problema de declasificación.

Si en el sistema los niveles de seguridad y permisos de acceso nunca cambian, se dice que tiene la propiedad de tranquilidad (*tranquility property*). Las operaciones que no cambian permisos de acceso se llaman tranquilas (*tranquil*).

Notas Finales

La exposición del modelo Bell-La Padula en el presente documento sigue de cerca la versión simplificada expuesta en [8], complementandose con otras fuentes [9, 10, 11] y tratando de seguir los reportes originales de Bell y La Padula. En particular, evita ahondar en detalles más finos de máquinas de estado presentes en el modelo original, necesarios para la prueba por inducción del teorema básico de seguridad.

Referencias

[1] David Elliot Bell and Leonard J. La Padula. Secure Computer Systems: Mathematical Foundations. Technical Report ESD-TR-73-278-I, MITRE Corporation, 1973.

[2] David Elliot Bell and Leonard. J. La Padula. Secure Computer Systems: A Mathematical Model. Technical Report ESD-TR-73-278-II, MITRE Corporation.

[3] David Elliott Bell. Secure Computer Systems: A Refinement of the Mathematical Model. Technical Report ESD-TR-73-278-III, MITRE Corp., 1974.

[4] David Elliot Bell and Leonard J. La Padula. Secure Computer System: Unified Exposition and MULTICS Interpretation. Technical Report ESD-TR-75-306, The MITRE Corporation, 1976.

[5] David. E. Bell. Looking back at the Bell-La Padula model. In *21st Annual Computer Security Applications Conference (ACSAC'05)*, pages 15 pp.–351, Dec 2005.

[6] Ravi S. Sandhu. Lattice-Based Access Control Models. *Computer*, 26(11):9–19, November 1993.

[7] John McLean. Reasoning About Security Models. In *1987 IEEE Symposium on Security and Privacy*, pages 123–123, April 1987.

[8] Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2003.

[9] D. Gollmann. *Computer Security*. Wiley, 2011.

[10] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, 2 edition, 2008.

[11] Henk C. A. Tilborg and Sushil Jajodia. *Encyclopedia of Cryptography and Security*. Springer Publishing Company, Incorporated, 2nd edition, 2011.

Apéndice

Solución 1. 1. El diagrama de Hasse del retículo formado por los niveles de seguridad y la relación dom se observa en Figura 3. Las flechas representan la relación de dominancia, y pueden pensarse como el sentido del flujo de información.

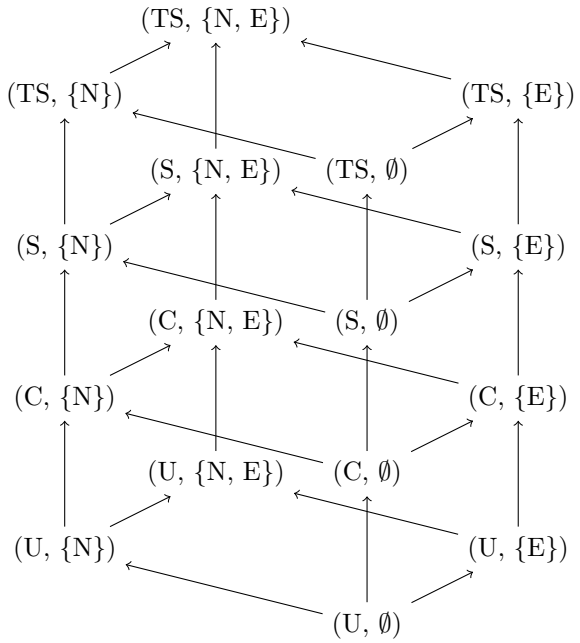

Figura 3: Retículo de niveles de seguridad.

2. Sí, L(Presidente) domina a L(Costo del p. nuclear) y L(Costo del ejercito), por lo que en base a la propiedad-ss, esta permitido.

3. No, el mayor no tiene acceso al número de unidades nucleares por que su nivel de seguridad no incluye la categoría N. En otras palabras, porque (C, {E}) ¬dom (C, {N}).

4. Sí, el nivel de seguridad del coronel domina al nivel de ambos objetos.

5. Sí, ya que (S, {N, E}) dom (S, {E}).

6. Ambos pueden, ya que L(Código Nuclear) = (TS, {N}) domina tanto al nivel del coronel (C, {N}) como al nivel del soldado (U, {N}), por lo que no contradicen la propiedad-*.

7. Bell-La Padula no garantiza integridad, solo confidencialidad. Esto significa que un sujeto con clasificación *Unclassified*, puede borrar (accidentalmente o no) información secreta. Para prevenir esto, a veces se usa una propiedad-* modificada que requiere $L(s) = L(o)$; es decir, todo sujeto puede escribir en su mismo nivel, pero no en mayores.