Problema Resuelto – Z

Maximiliano Cristiá

Ingeniería de Software 1
LCC – FCEIA – UNR
Rosario – Argentina
2024

1. Requerimientos

La política de seguridad informática que regula la confidencialidad en sistemas operativos se denomina *seguridad multi-nivel* (MLS[1]). Esta es la política de seguridad aplicada por el Departamento de Defensa (DoD) de los EE.UU., incluso antes de la aparición de las computadoras, a toda su información crítica o sensible, en particular aquella vinculada con los datos de inteligencia y militares.

Según la política de MLS:

- Cada documento y cada persona posee (o tiene asignada) una *clase de acceso* o *clase de seguridad*.
- Cada clase de acceso es un par ordenado de la forma (n, C) donde n se denomina *nivel de seguridad* y C es un conjunto de *categorías* (denominado *departamento*[2]).
- El nivel de seguridad es un elemento de un conjunto finito totalmente ordenado. El DoD utiliza el siguiente conjunto de etiquetas: *unclassified* < *confidential* < *secret* < *top secret*. Cuando esta política de seguridad se formaliza con algún lenguaje matemático se suele generalizar de forma tal que el nivel de seguridad es un número natural.
- Cada categoría presente en la segunda componente de una clase de seguridad pertenece a un cierto conjunto (de categorías) que podemos notar con $\mathbb{C}$ (es decir si C es la segunda componente de una clase de acceso entonces $C \subseteq \mathbb{C}$). Cada categoría refiere a un tema o asunto del cual trata el documento o sobre el cual la persona tiene un interés legítimo. Por ejemplo, en el ambiente militar de EE.UU. algunas categorías podrían ser: Iraq, Al-Qaeda, armas nucleares, misiles de largo alcance, etc. En la terminología del DoD a las categorías se las suele llamar *need-to-know*, es decir que representan temas o asuntos que la persona tiene la necesidad de saber o conocer. Usualmente $\mathbb{C}$ es un conjunto de mayor tamaño que el conjunto de niveles de seguridad.
- Una persona puede *leer* un documento sí y solo sí la clase de acceso de la persona *domina* a la clase de acceso del documento. La clase de acceso (n_1, C_1) domina a la clase de acceso (n_2, C_2) sí y solo sí $n_2 \leq n_1$ y $C_2 \subseteq C_1$. Es decir la relación *domina* define un orden parcial en el conjunto de clases de acceso.

Observar que la regla principal de la política MLS (la última) hace referencia únicamente al permiso de lectura. Esto es así debido a que la confidencialidad está relacionada con la

[1] 'multi-level security', en inglés.
[2] 'compartment', en inglés.

posibilidad de ver o leer información. Es decir, un dato se mantiene confidencial en tanto y en cuanto nadie no autorizado logre leerlo. En ese caso ese dato permanece secreto en el sentido de que solo lo saben o conocen aquellos que lo comparten. Esta regla de MLS se conoce, en inglés, como *no read-up*. Aun así, cuando esta política debe ser implementada en un sistema operativo se deben determinar reglas que regulen la escritura de objetos y la creación de nuevos objetos, como veremos un poco más adelante.

En MLS una vez que se asigna una clase de seguridad a un documento o a una persona, solo se puede modificar a través de un procedimiento administrativo complejo y altamente controlado. Por ejemplo, para aumentar la clase de seguridad de un oficial de inteligencia este será sometido a interrogatorios, averiguación de antecedentes, etc. con el fin de disminuir al mínimo las posibilidades de que traicione a la organización. En el mismo sentido los documentos deben pasar por una serie de controles antes de ser *desclasificados*. En particular debe pasar un cierto número de años (en ocasiones más de 50) para que ciertos documentos puedan ser vistos por el público en general.

La asignación de una clase de seguridad a un documento no queda a *discreción* de quien lo genera o ingresa al sistema. En este sentido la política MLS entra dentro de la categoría de *control de acceso obligatorio* (MAC[3]). Es decir, los usuarios ordinarios no pueden determinar ni modificar la política de seguridad.

2. La especificación Z – Modelo de Bell-LaPadula

El primer modelo de MLS fue propuesto en 1972 por dos investigadores norteamericanos de The Mitre Corp., Elliot Bell y Leonard LaPadula [1, 2]. Actualmente este modelo se conoce como modelo de Bell-LaPadula o mdelo BLP. BLP es la especificación de las llamadas al sistema de un sistema operativo antecesor a UNIX. Bell y LaPadula utilizaron una notación matemática simple pero ad-hoc para describir una máquina de estados cuyo espacio de estados está constituido por las estructuras de datos necesarias para imponer MLS y cuyas transiciones son las llamadas al sistema operativo. Además de MLS el modelo BLP incluye la especificación de listas de control de acceso (ACL). Las operaciones del sistema solo se pueden ejecutar si se verifican los controles ACL y MLS. Luego especificaron dos invariantes de estado que describen de manera concisa las propiedades claves para un sistema operativo que implemente MLS. Finalmente demostraron que la especificación de las transiciones preserva esos invariantes.

En lugar de utilizar la notación que usaron Bell y LaPadula nosotros utilizaremos Z. Además no vamos a especificar todas las llamadas al sistema que ellos especificaron sino solo dos que son muy representativas de las dificultades que se deben enfrentar cuando se trabaja con MLS. El control ACL tampoco lo especificaremos.

Comenzamos introduciendo los tipos básicos de nuestra especificación Z que son las categorías (CAT), niveles de seguridad (SL) y objetos (OBJ). Los objetos representan tanto a personas o procesos como a documentos o archivos.

$$[CAT, OBJ]$$

$$SL == \mathbb{N}$$

Los sujetos son un subconjunto no vacío de los objetos que representan a personas o procesos.

[3] 'mandatory access control', en inglés.

$$SUBJ : \mathbb{P}\, OBJ$$
$$SUBJ \neq \emptyset$$

Las clases de seguridad son pares ordenados de niveles de seguridad y conjuntos de categorías.

$$SC == SL \times \mathbb{P}\, CAT$$

También podemos definir la relación de orden parcial entre clases de acceso.

$$\_dominates\_ : SC \leftrightarrow SC$$
$$\forall a, b : SC \bullet a \; dominates \; b \Leftrightarrow b.1 \leq a.1 \land b.2 \subseteq a.2$$

Aunque BLP usa varios modos de acceso nosotros solo utilizaremos los dos modos de acceso básicos: lectura, llamado *rm*; y escritura, llamado *wm*.

$$AM ::= rm | wm$$

Ahora definimos el estado del sistema que se compone de:

- una función que asocia objetos con clases de acceso, *sc*. Notar que como los sujetos también son objetos *sc* se puede aplicar también a ellos.
- una función que asocia sujetos con los objetos que están accediendo y en qué modo, *oobj* (por 'Open OBJects')

SecState
$$sc : OBJ \nrightarrow SC$$
$$oobj : SUBJ \nrightarrow (OBJ \leftrightarrow AM)$$

Si pensamos en un sistema operativo tipo UNIX, los sujetos serían procesos, los objetos archivos, la variable *sc* los permisos guardados en los i-nodos, y la variable *oobj* los archivos que cada proceso tiene abiertos. Observar que si $s \in SUBJ$ entonces $oobj(s)$ es una relación binaria, o sea un conjunto de pares ordenados. Esto es así porque un sujeto o proceso puede tener abierto más de un objeto o archivo y cada archivo puede estar abierto en uno o dos modos. Entonces si s tiene abierto el objeto o en ambos modos tenemos $oobj(s) = \{o \mapsto wm, o \mapsto rm\}$.

El estado inicial es trivial.

ISecState
$$SecState$$
$$sc = \emptyset$$
$$oobj = \emptyset$$

Antes de formalizar los invariantes de estado que dieron originalmente Bell y LaPadula vamos a dar invariantes simples que complementan a los tipos que hemos definido.

InvType

SecState

$$\begin{aligned}&\operatorname{dom} oobj \subseteq SUBJ\\&\operatorname{dom} oobj \subseteq \operatorname{dom} sc\\&\forall x : \operatorname{ran} oobj \bullet \operatorname{dom} x \subseteq \operatorname{dom} sc\end{aligned}$$

Es decir, estos invariantes exigen que los sujetos que tienen abiertos objetos y esos mismos objetos sean sujetos y objetos del sistema.

La primera propiedad enunciada por Bell y LaPadula (en forma de invariante de estados) la llamaron *condición de seguridad*[4]. La condición de seguridad es la formalización directa de la regla de acceso a la información estipulada en MLS: una persona puede leer un documento sí y solo sí la clase de acceso de la persona domina a la clase de acceso del documento. Traducido al estado del sistema que hemos definido significa que si un sujeto tiene acceso de lectura a un objeto (i.e. $(s, o, rm) \in oobj$) es porque la clase de acceso de ese sujeto domina a la del objeto (i.e. $sc(s)$ *dominates* $sc(o)$).

InvSecCond

SecState

$$\forall s : \operatorname{dom} oobj;\ o : OBJ \mid (s, o, rm) \in oobj \bullet sc(s)\ \textit{dominates}\ sc(o)$$

La segunda propiedad exigida en BLP se llama *propiedad estrella* y se escribe *propiedad-**[5]. Esta propiedad no está enunciada en MLS y aparece en BLP pura y exclusivamente porque se trata de un sistema de cómputo donde existe la posibilidad de que caballos de Troya *escriban* datos. En efecto, si un sujeto tiene abiertos los objetos o_h y o_l, respectivamente, en modo de lectura y escritura[6], y son tales que $sc(o_h)$ *dominates* $sc(o_l)$, entonces le sería posible copiar los datos de o_h en o_l produciendo la desclasificación[7] no autorizada de la información contenida en o_h. Esto es posible porque un sistema operativo puede controlar a los procesos solo cuando efectúan llamadas al sistema cosa que no es necesaria cuando copian información entre áreas de memoria propias. Esta situación se representa gráficamente en la Figura 1. Allí se puede observar que la lectura del archivo o_h se efectúa por medio de la llamada al sistema read en tanto que la escritura en el archivo o_l se hace usando la llamada write; en ambos casos el sistema operativo puede controlar que todo esté de acuerdo a la política de seguridad. Sin embargo, la copia de los datos del arreglo h en el arreglo l (simbolizada por la línea gruesa) escapa al control del sistema operativo. Por lo tanto este no tiene forma de saber que el contenido de l es de clase de acceso $sc(o_h)$.

[4] 'security condition', en inglés.

[5] '*-property', en el original en inglés.

[6] Usualmente la letra 'h' se utiliza para denotar objetos con clase de acceso alta (por *high*, en inglés), mientras que 'l' se usa para objetos de clase de acceso baja (por *low*, en inglés). Luego del trabajo de Bell y LaPadula la mayoría de los investigadores del área comenzaron a proponer soluciones al problema de la confidencialidad donde las clases de acceso se simplifican considerando solo el nivel de seguridad, la relación de dominación se vuelve un orden total ($<$) y donde solo se consideran dos niveles, $L < H$, donde L es el nivel de la información *pública* y H es el de la información *secreta*. En este caso se habla más de niveles de seguridad que de clases de acceso. La generalización de estas soluciones a un retículo de clases de acceso en general es trivial.

[7] 'downgrading', en inglés.

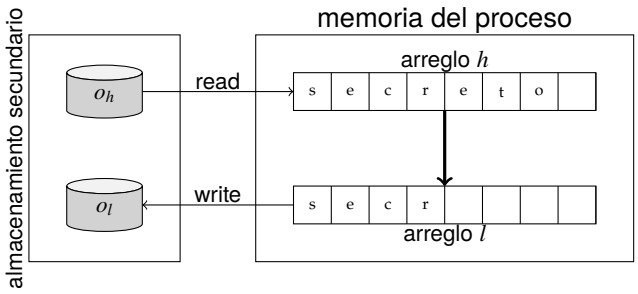

Figura 1: La desclasificación no autorizada de información es posible sin la propiedad-*

Entonces, la propiedad-* exige que, para cada sujeto, todos los objetos abiertos en modo de lectura tengan una clase de acceso dominada por la de cualquiera de los objetos abiertos en modo de escritura. En inglés esto se denomina *no write-down*; o sea que tenemos *no read-up* y *no write-down*.

InvStarProp

$$SecState$$

$$\forall s : \mathrm{dom}\, oobj;\ o_1, o_2 : OBJ \mid \{(s, o_1, wm), (s, o_2, rm)\} \subseteq oobj \bullet sc(o_1)\ dominates\ sc(o_2)$$

A continuación especificaremos las dos operaciones claves del sistema: solicitar acceso a un objeto en modo de lectura y en modo de escritura (es decir, las operaciones que representan la llamada al sistema UNIX open). Comenzamos por la apertura en modo *rm*. Observar que en la llamada al sistema el proceso que solicita la apertura es un parámetro de entrada implícito puesto que el sistema operativo está en condiciones de saber cuál es el proceso que hace la solicitud. Sin embargo, para simplificar la especificación Z incluimos $s?$ como variable de entrada.

OpenReadOk

$$\Delta SecState$$
$$s?, o? : OBJ$$

$$s? \in SUBJ$$
$$\{s?, o?\} \subseteq \mathrm{dom}\, sc$$
$$o? \notin \mathrm{dom}(oobj(s?) \rhd \{rm\})$$
$$sc(s?)\ dominates\ sc(o?)$$
$$\forall o : \mathrm{dom}(oobj(s?) \rhd \{wm\}) \bullet sc(o)\ dominates\ sc(o?)$$
$$oobj' = oobj \cup \{(s?, o?, rm)\}$$
$$sc' = sc$$

Es decir:

- $s?$ debe ser un sujeto
- $s?$ y $o?$ deben ser objetos existentes en el sistema porque deben tener una clase de seguridad asignada por sc

- $s?$ no tiene abierto a $o?$ en modo de lectura
- La clase de acceso de $s?$ domina a la de $o?$. Este predicado implementa la regla *no read-up* de MLS.
- La clase de acceso de $o?$ está dominada por la clase de acceso de todos los objetos que $s?$ tiene abiertos en modo de escritura. Este predicado apunta a controlar posibles *write-downs*.
- En ese caso $o?$ pasa a ser uno de los objetos que $s?$ tiene abiertos en modo de lectura

No especificaremos los casos de error aunque Bell y LaPadula lo hacen en el trabajo original. Asumimos que la especificación final es el esquema *OpenRead* que reúne la disyunción de *OpenReadOk* y los esquemas de error.

La siguiente operación es solicitar acceso en modo de escritura a un objeto. Si bien desde el punto de vista de MLS un sujeto podría escribir en objetos más altos, en BLP solo se permite la escritura en objetos al mismo nivel que el sujeto.

OpenWriteOk

$$\Delta SecState$$
$$s?, o? : OBJ$$

$$s? \in SUBJ$$
$$\{s?, o?\} \subseteq \mathrm{dom}\, sc$$
$$o? \notin \mathrm{dom}(oobj(s?) \rhd \{wm\})$$
$$sc(s?) = sc(o?)$$
$$\forall o : \mathrm{dom}(oobj(s?) \rhd \{rm\}) \bullet sc(o?)\ dominates\ sc(o)$$
$$oobj' = oobj \cup \{(s?, o?, wm)\}$$
$$sc' = sc$$

Luego de especificar las demás operaciones del sistema, Bell y LaPadula demuestran que todas ellas preservan la condición de seguridad y la propiedad-*. Básicamente demuestran teoremas como los que siguen:

theorem OpenReadSecCond
$InvSecCond \land OpenRead \Rightarrow InvSecCond'$

theorem OpenReadStarProp
$InvStarProp \land OpenRead \Rightarrow InvStarProp'$

theorem OpenWriteSecCond
$InvSecCond \land OpenWrite \Rightarrow InvSecCond'$

theorem OpenWriteStarProp
$InvStarProp \land OpenWrite \Rightarrow InvStarProp'$

a los cuales nosotros debemos agregar:

theorem OpenReadInvType
$InvType \land OpenRead \Rightarrow InvType'$

theorem OpenWriteInvType
$InvType \land OpenWrite \Rightarrow InvType'$

todos los cuales pueden demostrarse formalmente con una herramienta como Z/EVES. También es posible verificar automáticamente que las operaciones de BLP verifican las dos propiedades claves del modelo usando $\{log\}$ [3].

Dado que todas las operaciones preservan *InvSecCond* e *InvStarProp*, si el sistema parte de un estado en donde esos invariantes se verifican entonces el sistema será siempre seguro.

Para pensar, discutir y analizar

1. Especifique los esquemas de error de las operaciones *OpenRead* y *OpenWrite*.
2. Una lista de control de acceso (ACL) es una lista de pares ordenados de la forma (s, m) donde s es un sujeto y m es un modo de acceso. BLP asocia una ACL a cada objeto del sistema. De esta forma, el sujeto s puede acceder al objeto o en el modo m si la ACL de o contiene el par (s, m). Especifique las ACL y modifique las operaciones *OpenRead* y *OpenWrite* para que las tengan en cuenta.
3. En referencia al problema anterior especifique nuevos invariantes, si es necesario.
4. Especifique una operación que cierra un archivo abierto en modo de lectura.

Referencias

[1] D. Elliot Bell and Leonard LaPadula. Secure computer systems: Mathematical foundations. MTR 2547, The MITRE Corporation, May 1973.

[2] D. Elliot Bell and Leonard LaPadula. Secure computer systems: Mathematical model. ESD-TR 73-278, The MITRE Corporation, November 1973.

[3] Maximiliano Cristiá and Gianfranco Rossi. Automated proof of Bell-LaPadula security properties. *J. Autom. Reason.*, 65(4):463–478, 2021.