Grupo de Seguridad Informática

Fundamentos de la Seguridad Informática

Políticas y Modelos de Seguridad

Políticas de seguridad

- Qué es una política de seguridad?
- Qué reglas deben ser definidas?
- Para formular un política de seguridad se deben describir
  - Las entidades gobernadas por la política
  - La reglas que constituyen la política
- Qué tipo de definición?
  - Definición informal: documento en lenguaje natural
  - Definición formal: lógica y especificación de alto nivel del sistema

Modelos

- Han jugado un rol muy importante desde el comienzo de la actividad en el área
- Proceso de diseño y verificación
  - Especificación formal de la política que debe ser aplicada
  - Especificación de alto nivel del sistema de estudio y modelado de los mecanismos de seguridad
  - Verificación que la política es satisfecha por el modelo (prueba formal si alto nivel de aseguramiento es requerido)

Grupo de Seguridad Informática

Plan

- El modelo HRU (DAC)
- El modelo Bell-LaPadula (MAC)
  - Formalización de políticas de seguridad
  - Explicación del modelo, su alcance y limitaciones
- Otros Modelos de Seguridad
  - El modelo Chinese Wall – combina elementos de DAC y MAC
  - Modelo RBAC – es usualmente considerado un modelo neutral
  - El modelo Biba – relevante para integridad
  - El modelo de Information-Flow – generaliza las ideas subyacentes en MAC

Introducción

- Dos categorías principales:
  - Discretionary Access Control Models (DAC)
    - Definición [Bishop p.53] Si un usuario individual puede setear un mecanismo de control de acceso para permitir o denegar un acceso a un recurso, ese mecanismo constituye un *control de acceso discrecional (DAC*), también llamado un control de acceso basado en identidad (*IBAC*).
  - Mandatory Access Control Models (MAC)
    - Definición [Bishop p.53] Cuando un mecanismo de un sistema controla acceso a un objeto y un individuo no puede alterar ese acceso, entonces el control de acceso es mandatorio (MAC) (ocasionalmente llamado control de acceso basado en reglas)

DAC

- Políticas de seguridad DAC gobiernan el acceso de sujetos a objetos basándose en la identidad del sujeto, la identidad del objeto y los permisos de acceso
- Cuando un access request (AR) es sometido al sistema, el mecanismo de control de acceso verifica si existe un permiso que autorice el acceso
- Estos mecanismos son discrecionales ya que permiten que los sujetos puedan otorgarle a otros sujetos autorización de acceder sus propios objetos

DAC

- Ventajas:
  - Flexibilidad para especificar políticas
  - Provisto por todos los SOs y DBMS
- Desventajas:
  - No pueden controlar flujo de la información (ataques con Troyanos)

DAC – El modelo HRU

- El modelo Harrison-Ruzzo-Ullman (HRU) introdujo conceptos muy importantes:
  - La noción de *authorization systems*
  - La noción de *safety*

M.Harrison, W. Ruzzo, J. Ullman. Protection in Operating Systems. *Comm. of ACM* 19(8), August 1976.

El Modelo HRU

- Para describir el modelo HRU se requiere:
  - Un conjunto de sujetos *S*
  - Un conjunto de objetos *O*
  - Un conjunto *R* de permisos de acceso
  - Una matriz de control de acceso

$$M = (M_{so})_{s \in S,\, o \in O}$$

  - La entrada M_{so} es el subconjunto *R* que especifica los permisos que el sujeto *s* tiene sobre el objeto *o*

El modelo HRU – Operaciones Primitivas

- El modelo incluye 6 operaciones primitivas para manipular el conjunto de sujetos, el de objetos y la matriz de acceso:
  - **enter** *r* into M_{so}
  - **delete** *r* from M_{so}
  - **create subject** *s*
  - **delete subject** *s*
  - **create object** *o*
  - **delete object** *o*

El modelo HRU - Comandos

- Los comandos tienen la forma

command $c(x_1,.....,x_k)$

if r_1 in $M_{s1,o1}$ **and**

if r_2 in $M_{s2,o2}$ **and**

:

if r_m in $M_{sm,om}$

then $op_1,.....,op_n$

end

El modelo HRU - Comandos

- Los índices s_1,.....,s_m y o_1,.....,o_m son sujetos y objetos que ocurren en la lista de parámetros $c(x_1,.....,x_k)$
- La condición de los comandos chequean determinados permisos de acceso
- Si todas las condiciones se hacen verdaderas entonces se ejecuta la secuencia de operaciones básicas
- Cada comando contiene al menos una operación
- Los comandos que contienen exactamente una operación son llamados comandos *mono-operacionales*

El modelo HRU – Ejemplos de Comandos

command create_file (*s*,*f*)

create *f*

enter o into $M_{s,f}$

enter r into $M_{s,f}$

enter w into $M_{s,f}$

end

command grant_read (*s*,*p*,*f*)

if o in $M_{s,f}$

then enter r into $M_{p,f}$

end

El modelo HRU – Sistemas de protección

- Un sistema de protección se define como
  - Un conjunto finito de permisos
  - Un conjunto finito de comandos
- Un sistema de protección es un sistema de transición de estados

El modelo HRU – Estados

- Los efectos de un comando son registrados como cambios en la matriz de acceso (usualmente denotada *M*')
- La matriz entonces describe el estado del sistema de protección
- Qué representa el estado del sistema de protección?
  - El estado de un sistema es la colección de los valores corrientes de las celdas de memoria, la memoria secundaria y los registros y otros componentes del sistema
  - El estado del sistema de protección es el subconjunto de esa colección que está asociado a las direcciones de los permisos de acceso, representado por la matriz de acceso

El modelo HRU – Estados

- **Definición**. Un estado, es decir, una matriz de acceso, se dice que *gotea (leaks)* el permiso *r* si existe un comando *c* que agrega el permiso *r* en una entrada de la matriz de acceso que anteriormente no contenía ese permiso. Más formalmente, existen *s* y *o* tales que $r \notin M_{so}$ y, luego de la ejecución de *c*, $r \in M'_{so}$.
- Nota: El hecho de que un permiso gotee no es necesariamente malo, muchos sistemas permiten a sujetos darle acceso a otros sujetos

Grupo de Seguridad Informática

El modelo HRU – Estados seguros

- Qué se entiende por un estado seguro?
- Definición 1: “acceso a los recursos sin el consentimiento del propietario es imposible” [HRU76]
- Definición 2: “un usuario debería ser capaz de saber si la acción que va a efectuar (por ejemplo, otorgar un permiso) puede provocar que el mismo gotee hacia sujetos no autorizados” [HRU76]

El modelo HRU – Safety

- El problema que motiva la introducción de este concepto puede ser descrito de la siguiente forma:

  "*Suponer que un sujeto s planea darle al sujeto s' el permiso r sobre el objeto o. La pregunta obvia es si la matriz de acceso corriente, con r agregado en la entrada (s',o), es tal que ese permiso r podría a su vez posibilitar la entrada de algún otro permiso no existente.*"

El modelo HRU – Un ejemplo de sistema de protección unsafe

- Asuma un sistema de protección con los siguientes dos comandos:

command grant_execute (*s*,*p*,*f*)
if o in $M_{s,f}$
then enter x into $M_{p,f}$
end

command modify_own_right (*s*,*f*)
if x in $M_{s,f}$
then enter w into $M_{s,f}$
end

Grupo de Seguridad Informática

El modelo HRU – Un ejemplo de sistema de protección inseguro

- Suponer que el usuario Juan desarrolló un programa que él desea sea ejecutado por otros usuarios pero no puedan modificarlo
- El sistema anterior no es seguro respecto a esta política, considere la siguiente secuencia de comandos:

  - Juan: grant_execute (Juan, José, P_1)

  - José: modify_own_right (José, P_1)

  resulta en una matriz de acceso donde la entrada $M_{José,P1}$ contiene el permiso de acceso w

El modelo HRU – Safety

- **Definición**. Dado un sistema de protección y un permiso *r*, la configuración inicial Q_0 es unsafe para *r* (o gotea *r*) si existe una configuración *Q* y un comando c tales que
  - *Q* es alcanzable desde Q_0
  - c gotea *r* desde *Q*

Q_0 es safe para *r* si Q_0 no es unsafe para *r*.

- **Definición alternativa**. Un estado de un sistema de protección, o sea, su matriz *M*, es safe respecto al permiso *r* si no existe secuencia de comandos que puedan transformar *M* en un estado que gotee *r*.
- **Teorema**. Dada una matriz de acceso *M* y un permiso *r*, verificar la seguridad de *M* respecto a *r* es un problema indecidible

El modelo HRU - Conclusiones

- Los resultados sobre la decibilidad del problema de safety ilustran un principio importante, el *principio de economía de mecanismos*
  - Si uno diseña sistemas complejos que pueden solamente ser descriptos por modelos complejos, resulta muy difícil encontrar pruebas de safety de los mismos
  - en el peor caso (indecibilidad), no existe un algoritmo universal que verifica la seguridad del mismo para todas las posibles instancias del problema

Otros Modelos Teóricos

- El modelo take-grant

(A. Jones, R. Lipton, and L. Snyder)

- El modelo Typed Access Matrix

(R. Sandhu)

Otros Modelos

- Los modelos DAC han sido extensivamente investigados en el área de DBMS
- El primer modelo para bases de datos relacionales fue desarrollado por Grifiths y Wide
- Varias extensiones a ese modelo han sido desarrolladas

DAC – Características adicionales

- Flexibilidad es incrementada soportando diferentes clases de permisos
  - Positivos vs. negativos
  - Implícitos vs. explícitos
  - Basados en contenido

Permisos basados en contenido

- Este tipo de control de acceso condiciona el acceso a un objeto basado en el contenido del mismo
- Este tipo de permisos es principalmente relevante para sistemas de base de datos
- Ejemplo: en un RDBMS que implementa control de acceso basado en contenidos es posible autorizar un sujeto a acceder información solamente de aquellos empleados cuyo salario no es mayor que una suma dada

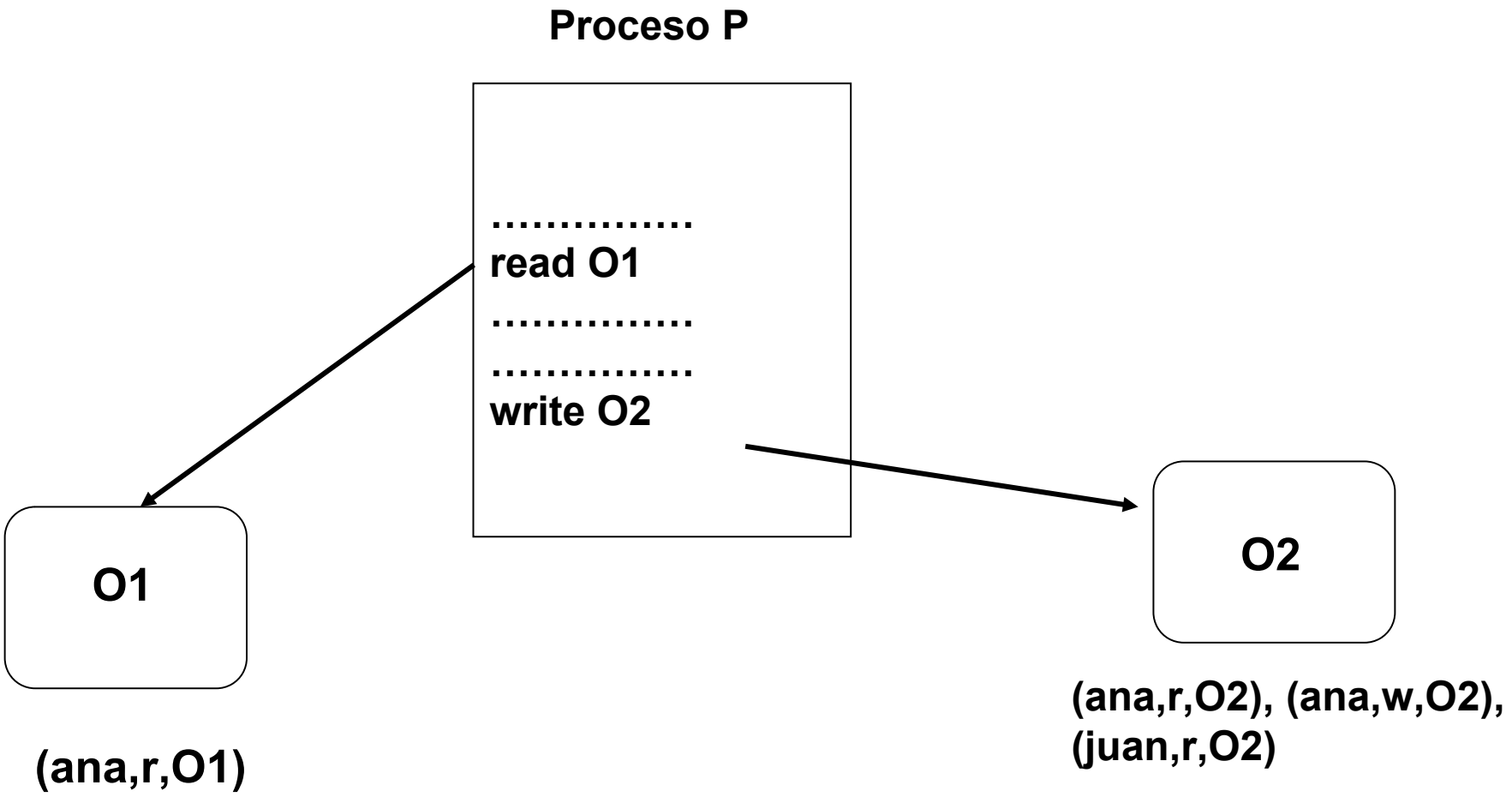
Permisos basados en contenido

- Son dos los enfoques más comunes para enforzar control de acceso basado en contenidos en DBMS:
  - Associando un predicado con el permiso
  - Definiendo una *vista* que selecciona aquellos objetos cuyo contenido satisface una condición dada, y entonces otorgando permisos sobre la vista en vez de hacerlo sobre los objetos básicos

Modelos DAC - DBMS vs SO

- Mayor número de objetos a ser protegidos
- Diferentes niveles de granularidad (relaciones, tuplas, atributos simples)
- Protección de estructuras lógicas (relaciones, vistas) en lugar de recursos reales (files)
- Diferentes niveles de arquitectura con diferentes requerimietnos de protección
- Relevancia no solamente de la representación física de los datos, también de su semántica

Troyanos (Trojan Horse)



Troyanos

- Los modelos DAC no tienen capacidad para proteger datos contra Troyanos embebidos en programas de una aplicación
- Los modelos MAC fueron desarrollados para prevenir este tipo de acceso ilegal

MAC

- MAC especifica el acceso que los sujetos tienen sobre objetos basado en la clasificación de seguridad que se hace de esos sujetos y objetos
- Este tipo de seguridad también es conocido como *multilevel security*
- Los sistemas de base de datos que satisfacen seguridad multinivel son llamados *multilevel secure database management systems* (MLS/DBMSs)
- La mayoría de los MLS/DBMSs han sido diseñados usando el modelo de Bell y LaPadula (BLP)

El Modelo BLP

- Uno de los modelos de seguridad más difundidos
- Seguridad de SO multi-usuario
- Sistemas que procesan información clasificada de distintos niveles deberían implementar MLS
- Los usuarios sólo deberían poder acceder a la información que están autorizados (*clearance*)

El Modelo BLP

- BLP se formuló como un modelo de Máquinas de Estado que captura aspectos de confidencialidad
- Permisos de acceso se definen usando una matriz de control de acceso y etiquetas de seguridad
- Las políticas establecen que la información no puede fluir hacia niveles de seguridad inferiores a los del repositorio origen
- El modelo sólo considera el flujo que ocurre cuando un subject observa o altera un objeto

El Estado

- Se desea utilizar el estado del sistema para verificar su seguridad, entonces el conjunto de estados del modelo debe capturar todas las instancias de sujetos que están accediendo a objetos y todos los permisos especificados
- Conjuntos base
  - Conjunto S de sujetos
  - Conjunto O de objetos
  - Conjunto A = {read,write,execute, append} de operaciones de acceso
  - Conjunto L de etiquetas de seguridad, con un orden parcial ≤

El Estado

- Componentes
  - Tabla de operaciones de acceso: b : B = [S x O x A]
  - Matriz de permisos: m : M = (M_{so})
  - Funciones de asignación de niveles de seguridad
    - F_s : S -> L (nivel maximal de seg. de sujetos)
    - F_c: S -> L (nivel de seguridad corriente de sujetos)
    - F_o: O -> L (clasificación de los objetos)
    - f: F = F_s x F_c x F_o
- El estado = (b,m,f) : B x M x F

Seguridad simple

- BLP define seguridad como una propiedad que cumplen los estados
- MLS permite a un sujeto leer un objeto sólo si el nivel de seguridad del sujeto domina al del objeto
  - ***Propiedad de Seguridad Simple (ss)***: Un estado (b,m,f) satisface la propiedad ss, si para cada tupla (s,o,a) de b donde la operación a es *read o write* se cumple $F_o(o) \leq F_s(s)$
- Esta propiedad captura la política de confidencialidad no read-up

Desclasificación

- Sistemas donde sujetos son procesos
  - No tienen memoria, pero
  - Tienen acceso a objetos de memoria
  - Pueden actuar como canales leyendo un objeto y transfiriendo la información a otro objeto
- Un atacante puede insertar un troyano en un objeto de alto nivel de seguridad y copiar información de objetos de alto nivel en objetos de inferior nivel de seguridad

La propiedad *

- BLP incluye una propiedad de no escritura hacia abajo pero que refiere al nivel corriente de seguridad del sujeto
  - ***La propiedad ****: Un estado (b,m,f) satisface esta propiedad si para cada tupla (s,o,a) de b donde la operación a es *write* o *append* el nivel corriente de seguridad del sujeto s es dominado por la clasificación de o, es decir se cumple que $F_c(s) \leq F_o(o)$.
  - Mas aún, si existe una tupla (s,o,a) de b donde la operación a es *write* o *append* , entonces de debe cumplir que $F_o(o') \leq F_o(o)$ para o' en (s,o',a') y a' es *read* o *write*.

Estado y Transición Seguros

- Un *estado* (b,m,f) se dice que *es seguro* si satisface las propiedades ss, * y sd
- Una transición del estado $s_1 = (b_1,m_1,f_1)$ al estado $s_2 = (b_2,m_2,f_2)$ es segura si los dos estados son seguros
- Transiciones deben preservar las propiedades de seguridad
- Ejemplo: La propiedad ss es preservada por la transición sii
  - cada $(s,o,a) \in \{b_2 - b_1\}$ satisface ss respecto a f_2
  - si $(s,o,a) \in b_1$ no satisface ss respecto a f_2, entonces $(s,o,a) \notin b_2$

Tranquility

- Cuestionamiento de McLean al modelo BLP: sistema con una transición que
  - Asigna a sujetos y objetos el nivel mínimo de seguridad
  - Asigna todos los permisos a cada entrada de la matriz de control de acceso
- Esta transición es segura según la definición de BLP, ... realmente lo es?
  - En contra de BLP: un sistema que puede degenerarse así no es seguro (McLean)
  - A favor de BLP: si es un requerimiento del usuario la transición debe ser admitida, sino no lo debe ser (Bell)

Tranquility

- El punto central de esta discusión es una transición de estado que cambia los permisos de acceso
- Estos cambios son admitidos en el marco general de BLP
- Los autores consideraron sistemas donde los permisos de acceso son invariantes
- La propiedad de que los permisos de acceso y los niveles de seguridad nunca son modificados es llamada *Tranquility*

La interpretación Multics de BLP

- **Multics**: proyecto de investigación cuyo objetivo era desarrollar un SO multi-usuario seguro y confiable
- Motivó una gran cantidad de trabajo en seguridad
- Precursor de Unix: balance entre seguridad y usabilidad
- Modelo de seguridad usado para diseñar un SO seguro: definición de Multics consistente con BLP

Bibliografía y Referencias

- **D. Gollman,** *Computer Security***,** Wiley, 2006.
- **E. Bertino**, *Notes of Information Security course*, Purdue University, 2005.
- **M.A. Harrison, W.L. Ruzzo, J.D. Ullman**, *Protection in Operating Systems*, Comm. ACM, 1976.
- **D.E. Bell, L. LaPadula**, *Secure Computer Systems: Mathematical Foundations*, MTR-2457, Vol. 1, The MITRE Corporation, 1973.
- **D.E. Bell, L. LaPadula**, *Secure Computer Systems: Unified Exposition and Multics Interpretation*, MTR-2997, The MITRE Corporation, 1975.

Bibliografía y Referencias

- **K.J. Biba**, *Integrity Considerations for Secure Computer Systems*. MITRE report TR-3153, 1977.
- **D.F.C Brewer, M.J. Nash**, *The Chinese Wall Security Policy*, Proc. IEEE Symp. Research in Security and Privacy, 1989.
- **D.R. Clark, D.R. Wilson**, *A comparison of commercial and military computer security policies*, Proc. IEEE Symp. Research in Security and Privacy, 1987.
- **R.S. Sandhu,** *Lattice-Based Access Control Models*, IEEE Computer, 1993.