Grupo de Seguridad Informática

Seguridad Informática

Identificación, Autenticación, Autorización

GSI - Facultad de Ingeniería

Plan

- Identificación y Autenticación
- Gestión de passwords
- Control de acceso
- Seguridad multinivel

Usuarios y Autenticación

- Existen dos razones para autenticar a los usuarios de un sistema:
  - La identidad del usuario es un parámetro para la decisión de control de acceso
  - La identidad del usuario es registrada cuando se hace el loggin de eventos relevantes para la seguriadad en la auditoría
- No es siempre necesario (o deseable) basar control de acceso en identidad de usuarios, aunque sí es esencial loguear identidades para poder auditar

Usuarios y Autenticación (II)

- Cuando un usuario se conecta a un sistema de computadoras el mismo debe proveer
  - *User name* – este paso se llama *identificación*
  - *Password* – este paso se llama *autenticación*
- Autenticación: es el proceso de verificar una (pretendida) identidad

Verificación de Identidades

- Una o más de los siguientes
  - Algo que se sabe (*ej.* password)
  - Algo que se tiene (*ej.* badge, token, smart card)
  - Algo que se es (*ej.* Huella digitales, ADN, iris)
  - Donde se está (*ej*. Usando una terminal particular)

Proceso de Autenticación

- Consiste de varios pasos:
  - Obtener la información de autenticación de una entidad
  - Analizar los datos
  - Determinar si la información de autenticación es efectivamente asociada a la entidad

Sistema de Autenticación

- Tupla (*A*, *C*, *F*, *L*, *S*)
  - *A* : información que prueba la identidad
  - *C* : información almacenada en la computadora y que es usada para validar la información de autenticación
  - *F* : función de complementación *f* : *A* -> *C*
  - *L* : funciones que prueban la identidad
  - *S* : funciones que le permiten a la entidad crear o alterar información en *A* o *C*

Ejemplo

- Sistema de password, con passwords almacenadas en claro y en línea
  - *A* conjunto de caracteres
  - *C* = *A*
  - *F* = función identidad { *I* }
  - *L* = función de testeo de igualdad{ *eq* }
  - *S* = función para setear/cambiar la password

Mecanismos de autenticación

- Passwords
- Desafío-Respuesta
- Mecanismos alternativos
- Métodos múltiples

Passwords

- Secuencia de caracteres
  - Ejemplos: 10 dígitos, un string de letras, *etc*.
  - Generado randómicamente, por el usuario, por la computadora usando input del usuario
- Secuencia de palabras
  - Ejemplos: pass-phrases
    - Una pass-phrase *es* una secuencia de caracteres que es demasiado larga para ser una password y que es por lo tanto asociada a una password virtual más corta por el sistema de passwords
- Algoritmos
  - *challenge-response*, *one-time passwords*

Almacenamiento

- Texto claro
  - Si el archivo de passwords (password file) es comprometido, *todas* las passwords son reveladas
- Archivo cifrado
  - Requiere tener claves de encriptado/desencriptado en memoria
  - Reduce al problema previo
- Almacenar *one-way hash* de la password
  - Si el archivo es leído, el atacante de todas formas necesita adivinar las passwords o invertir el hash

Autenticación basada en Passwords

- *Una password* es información que confirma la identidad de una entidad asociada
- Cómo pueden ser protegidas las passwords?
- Una solución: *one-way hashing*
  - La password de un usuario es encriptada y luego almacenada. La misma nunca es desencriptada.
  - Debe ser difícil para un atacante invertir la password almacenada.
  - Un usuario A puede intentar adivinar la password de otro usuario, B, y así usurpar la *identidad* de B. (próxima diapo)

Ejemplo

- Función estándar de hash del sistema UNIX
  - Hashea passwords en un string de 11 caracteres usando una de 4096 funciones de hash
- Sistema de autenticación:
  - *A* = { strings de 8 caracteres o menos }
  - *C* = { 2 char (hash id) || 11 char (hash) }
    - El 2 char identifica la función de hash usada
  - *F* = { 4096 versiones de DES modificado }
  - *L* = { *login*, *su*, … }
  - *S* = { *passwd*, *nispasswd*, *passwd+*, … }

Análisis de un ataque de usurpación de identidad

- Objetivo: encontrar $a \in A$ tal que:
  - Para algún $f \in F$, $f(a) = c \in C$
  - c está asociada a la identidad dada
- Dos formas de determinar si a satisface los requerimientos:
  - Enfoque directo: como indicado arriba – es posible si C es conocido por el atacante
  - Enfoque indirecto: como $I(a)$ puede ser exitosa sii $f(a) = c \in C$ para algún c asociado con una entidad, computar $I(a)$

Previniendo Ataques

- Esconder uno de *a*, *f*, o *c*
  - Previene ataques obvios
  - Ejemplo: UNIX/Linux shadow password files
    - Esconde *c*'s
      - Solamente puede ser accedido por el super-usuario (se usa control de acceso)
- Bloquear acceso a toda $l \in L$ o resultado de $l(a)$
  - Previene que el atacante pueda enterarse de si la password ha sido adivinada
  - Ejemplo: prevenir logins a una determinada cuenta desde la red
    - Previene conocer resultados de *l* (o acceder a *l*)

Ataques de Diccionario

- Ensayo-y-error a partir de una lista de passwords potenciales
  - Tipo 1: el atacante conoce *A*, *f*, *c*
    - También conocido como *Off-line*: el atacante conoce *f y c*s, y repetidamente trata diferentes ensayos $g \in A$ hasta que la lista se acaba o la password es adivinada
  - Tipo 2: el atacante conoce *A*, *l*
    - También conocido como *On-line*: el atacante tiene acceso a las funciones en *L* y ensaya con adivinanzas *g* hasta que algún *l*(*g*) es exitoso
    - Ejemplos: tratar de loguearse adivinando una password

Defendiéndose de Password Guessing

- El objetivo es maximizar el tiempo necesario para adivinar una password
- Fórmula de Anderson:

  - *P* : probabilidad de adivinar una password en un período de tiempo especificado
  - *G* : número de adivinanzas en 1 unidad de tiempo
  - *T* : número de unidades de tiempo
  - *N* : número de passwords posibles (|*A*|)

  Entonces $P \geq TG/N$

Ejemplo de Uso de la Fórmula de Anderson

- Considerar el caso de un PIN de 4 dígitos
- Suponer que el número de passwords posibles (PINs) es $N=10^4$ (asumiendo que los dígitos 0-9 están permitidos en cada una de las 4 posiciones del PIN)
- Asumir que un atacante puede hacer G=10.000 por segundo en un ataque offline
- Cuánto tiempo insumiría poder adivinar un PIN específico con certeza absoluta?
- $P \geq TG/N$, o, $T \leq PN/G$ = (1.0 *10.000)/10.000= 1

Alternativas: Selección de Password

- Selección Randómica
  - Cualquier password de *A* con igual probabilidad de ser seleccionada
  - Esas passwords son difíciles de recordar por parte de los usuarios, especialmente cuando cuando los mismos tienen múltiples passwords randómicamente selccionadas
- Passwords pronunciables
- Selección de passwords por el usuario

Passwords Pronunciables

- Generación randómica de fonemas
  - Un fonema es una unidad de sonido, ej. *cv*, *vc*, *cvc*, *vcv* *donde*
    - *c* es una consonante
    - *v* es una vocal
  - Ejemplos: helgoret, juttelon son pronunciables; przbqxdfl, zxrptglfn no son pronunciables
- Problema: el número de passwords pronunciables de largo *n* es considerablemente menor que el número de passwords randómicas de largo *n*

Selección por el Usuario

- Problema: la gente elige passwords fácilmente adivinables
  - Basadas en nombres de cuentas, usuarios, nombres de computadoras, lugares
  - Palabras de diccionario(invertidas, con capitalizaciones extrañas, caracteres de control, …)
  - Demasiado cortas, solamente dígitos. Solamente letras
  - Matrículas, acrónimos,…
  - Características personales, mascotas, sobrenombres, …)

Seleccionando Buenas Passwords

- Buenas passwords pueden ser construídas de diversas formas
  - Una password que contenga al menos un dígito, una letra, un símbolo de puntuación, y un carácter de control, es generalmente una buena password
- “LlMm*2^Ap”
  - Letras elegidas a partir de nombres de miembros de 2 familias
- “3T5d2P6”
  - Número de palabras seguido de primera letra y número de letras de cada palabra de una pass-phrase (por ejemplo una película preferida)

Salting

- Objetivo: ataques de diccionario lentos diseñados para encontrar una password de cualquier usuario (opuesto a un usuario en particular)
- Método: perturbar la función de hash:
  - Parámetro controla qué función es usada
  - Parámetro difiere para cada password
  - Para determinar si el string *s* es la password para cualquiera de un conjunto *n de usuarios*, el atacante tiene que ejecutar *n* complementaciones, cada una de las cuales genera a su vez un complemento diferente

Expiración de Password

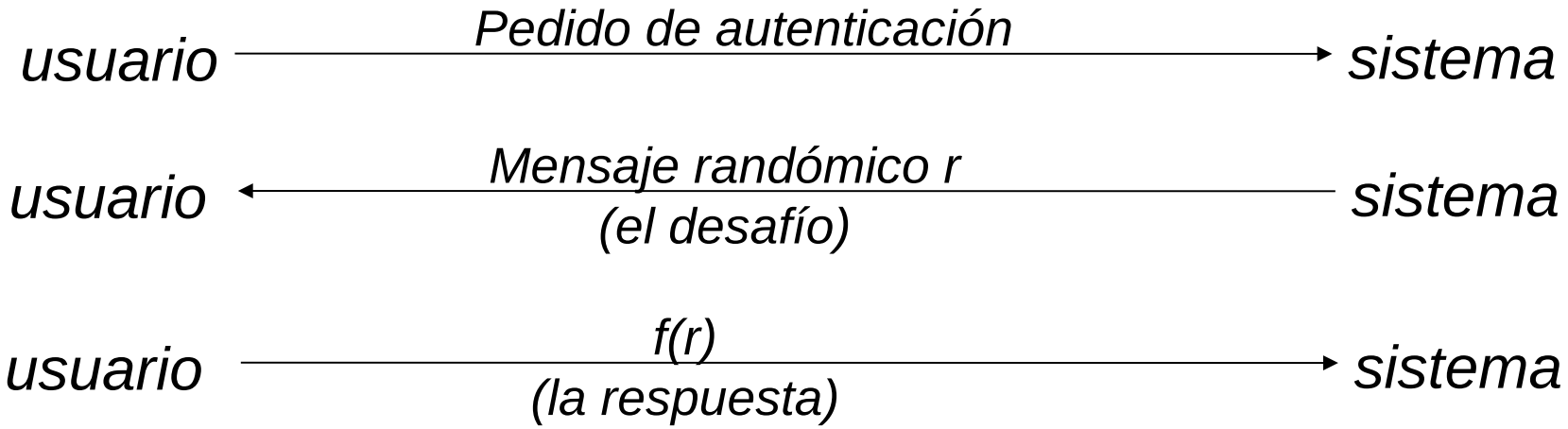
- Forzar a los usuarios a cambiar passwords luego de un cierto tiempo
  - Cómo forzar a que no se repitan passwords?
    - Registrar passwords previas
    - Bloquear cambios por un cierto período de tiempo
  - Darle a los usuarios tiempo para pensar buenas passwords
    - No forzarlos a cambiar antes de que puedan loguearse
    - Advertirlos acerca de los tiempos de expiración

Desafío-Respuesta

- Las passwords tienen el problema intrínseco de que son *reusables*
- Si un atacante se apodera de una password, luego puede hacer un *replay* de la misma
- Una alternativa es autenticar de forma que la password transmitida cambie cada vez
- Sea *u* un usuario que desea autenticarse ante un sistema *S,* donde *u* y *S* se han puesto de acuerdo en una función secreta *f*. Un sistema de autenticación *desafío-respuesta* es uno en el que *S* envía un mensaje randómico *m* (el desafío) a *u*, y *u* replica con la transformación *r* = *f*(*m*) (la respuesta). *S* entonces valida *r* computándolo por su lado.

Desafío-Respuesta

El usuario y el sistema comparten una función secreta *f* (de hecho, *f* puede ser una función conocida con parámetros desconocidos, como una clave criptográfica)



Desafío-Respuesta
Pass Algorithms

- Desafío-respuesta con la función *f* como secreto
  - Ejemplo:
    - El desafío es un string randómico de caracteres como “abcdefg”, “ageksido”
    - La respuesta es alguna función de ese string como “bdf”, “gkio”
    - El algoritmo es tomar letras, saltando una, a partir de la segunda
  - Se puede alterar el algoritmo basándose en información suplementaria
    - Network connection es como arriba, dial-up puede requerir “aceg”, “aesd”
  - Usualmente usado en conjunción con password fija y reusable

Desafío-Respuesta

Enfoques basados en claves públicas criptográficas

- Objetivo: *A* identifica a *B* verificando si *B* posee la clave secreta k_B que machea la clave pública K_B
- Hipótesis: *A* elige un desafío randómico (nonce) r_A. *B* usa su propio nonce r_B. *B* aplica su sistema de clave pública para autenticación
- Secuencia de mensajes:

1. $A \rightarrow B: r_A.$
2. $B \rightarrow A: r_B, < r_A, r_B > k_B.$

Desafío-Respuesta
Enfoques basados en claves públicas criptográficas

- Pasos:
  - A envía su desafío randómico r_A a B
  - B toma un nonce r_B fresco y firma el par de nonces
  - La firma $< r_A, r_B > k_B$ es enviada a A quien verifica su validez en la forma usual

One-Time Passwords

- Passwords que pueden ser usadas exactamente una vez
  - Luego de su uso es inmediatamente invalidada
- Problemas
  - Exige alta sincronización del usuario con el sistema
  - Generación de buenas passwords randómicas
  - Problema de distribución de password

S/Key

- Esquema de one-time password basado en ideas de L. Lamport
- *h* es una función de hash one-way (SHA-3, por ejemplo)
- El usuario elige la semilla inicial *k*
- El generador de claves calcula:

$$h(k) = k_1,\ h(k_1) = k_2,\ \ldots,\ h(k_{n-1}) = k_n$$

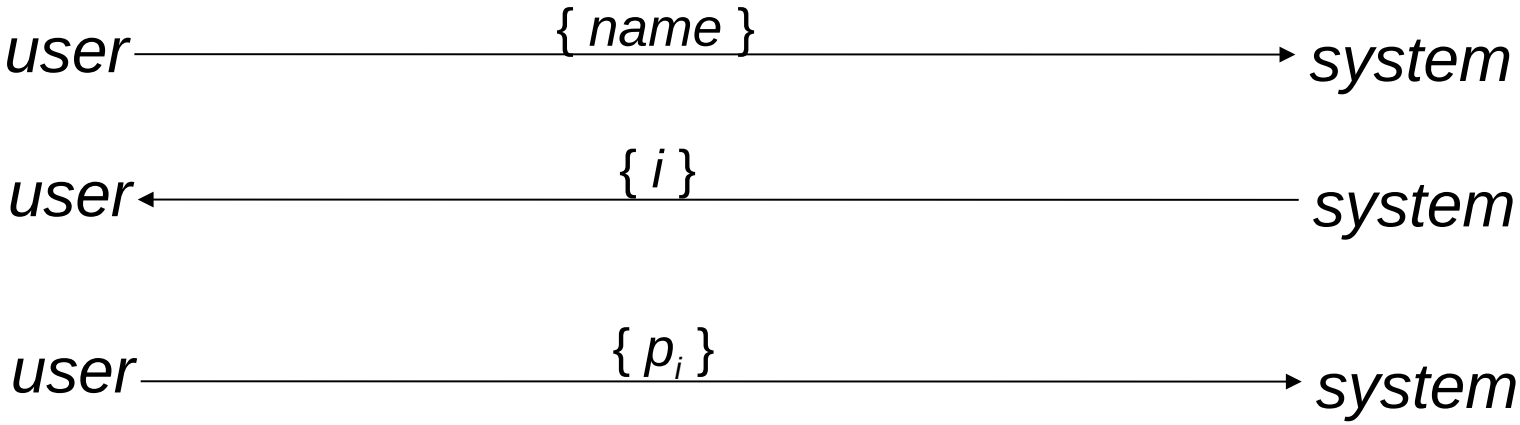
- Las passwords se definen en orden inverso: $p_1 = k_n$, $p_2 = k_{n-1}$, …, $p_{n-1} = k_2$, $p_n = k_1$

S/Key

- Suponer que un atacante intercepta p_i
- Como $p_i = k_{n-i+1}$, $p_{i+1} = k_{n-i}$, y $h(k_{n-i}) = k_{n-i+1}$, entonces $h(p_{i+1}) = p_i$
- Entonces, para que el atacante pueda adivinar p_{i+1} a partir de p_i tendría que poder invertir h; como h es una función one-way function, esto no es posible

Protocolo S/Key

El sistema almacena el máximo número de autenticaciones n, el número de la próxima autenticación i, y la última correctamente provista password p_{i-1}.



El sistema computa $h(p_i) = h(k_{n-i+1}) = k_{n-i+2} = p_{i-1}$. Si machea con lo que está almacenado, el sistema remplaza p_{i-1} con p_i e incrementa i.

Mecanismos Alternativos

- Opciones alternativas de autenticación
  - Algo que se sabe
  - Algo que se tiene
  - Quién se es
  - Qué se hace
  - Donde se está

Algo que se sabe

- El usuario tiene que conocer un secreto para ser autenticado
  - Password
  - PIN
  - Información personal
- Se autentica a quien conoce el secreto
- La transferencia del secreto no deja trazas
- Dificil probar “impersonation”

Algo que se tiene

- El usuario debe presentar un token físico para ser autenticado
  - Llave
  - Tarjeta o tag de identificación
  - Tarjetas inteligentes
- Usualmente los token son usados en combinación con un secreto
- Información sensible puede ser obtenida o transferida

Quién se es

- Esquemas biométricos de autenticación
  - Huellas dactilares
  - Patrones de iris
  - Geometría de la mano
  - ADN
- Características y procedimientos de la autenticación basada en huellas dactilares

Biometría

- Medida automática de características biológicas/comportamentales que identifican a una persona
  - Huellas digitales: técnicas ópticas o eléctricas
    - Mapea huellas a un grafo, luego compara con base de datos
    - Medida imprecisa, por lo tanto se usan algoritmos de macheo aproximado
  - Voces: verificación/reconocimiento de habla
    - Verification: usa técnicas estadísticas para testear la hipótesis de que el que habla es quien dice ser (depende del que habla)
    - Reconocimiento: chequea contenido de respuestas (independiente de quien habla)

Otras Características

- Se pueden usar otras características
  - Ojos: patrones únicos en iris
    - Patrones de medida determinan si las diferencias son randómicas o correlacionan imágenes usando tests estadísticos
  - Rostros: imagen, o características específicas como distancia de la nariz a la mandíbula
  - Dinámica de tecleo: se piensa que es única por persona
    - Intervalos entre teclas, presion, duración de tecleo, donde es golpeada la tecla
    - Se usan tests estadísticos

Algo que se hace

- La gente desarrolla mecánicamente tareas que son repetibles y específicas al individuo
  - Firmas manuscritas
  - Fácil falsificación
- Firmas sobre un dispositivo que mida atributos como velocidad y presión de escritura
- En teclados: velocidad e intervalos de tipeado
- Nuevamente el problema de falsos positivos y negativos

Dónde se está

- Cuando el usuario se loguea el sistema puede tomar también en cuenta dónde se encuentra
  - SO que verifican que un usuario sólo se loguee desde una cierta terminal
- Si se necesita localidad geográfica precisa: GPS?
- Identificar el lugar desde que un usuario se autentica puede ayudad a resolver disputas sobre la verdadera identidad del mismo

Métodos múltiples

- Ejemplo: “donde estás” también requiere de la entidad “algo que tenga”, como el GPS
- Se pueden asignar diferentes métodos a tareas diferentes
  - Como los usuario ejecutan cada vez tareas más sensible, se deben autenticar de formas más variadas
  - Pluggable Authentication Modules

Ingeniería Social

Riesgos:

- Explota vulnerabilidades del factor humano.
- Extrae información (pwds, datos personales, etc.) mediante falsos pretextos.
- Dispositivos de autenticación (generador de tokens, etc.) no siempre son aplicables o viables

Contramedidas:

- Administradores de sistemas y usuarios concientizados ante amenazas y riesgos.
- Política de seguridad de la información

Resumen

- Autenticación no es (solamente) criptografía
  - Hay que considerar los componentes del sistema
- Las passwords constituyen un mecanismo duradero
  - Proveen las bases para la mayoría de las formas de autenticación
- Los protocolos son muy importantes
  - Pueden complicar el encubrimiento
- Métodos de Autenticación pueden ser combinados

Algunas conclusiones

- Una password o secreto no autentica a una persona, una autenticación exitosa sólo garantiza que la persona conoce el secreto
- No hay forma de diferenciar un usuario legítimo de un intruso que ha podido acceder al secreto

Control de Acceso

Autorización

Es el proceso que determina (luego de su autenticación) a qué recursos de un sistema tiene acceso una identidad

Plan

- Operaciones y Modos de Acceso
- Estructuras de Control de Acceso
- Fundamentos de la Seguridad Multiniveles

Control de Acceso

- Control de acceso = Autenticación + Autorización
- Acceso: sujeto, objeto, operación
- Si **s** es una sentencia,
  - Autenticación responde la pregunta: *quien dijo **s**?*
- Si **o** es un objeto,
  - Autorización responde la pregunta: *quién tiene acceso a **o**?*

Grupo de Seguridad Informática

Sujetos, Principals, Objetos

- IBAC
  - Sujetos actúan en representación de usuarios humanos, o principals
  - Acceso está basado en la asociación entre la identidad del usuario y el sujeto
- Alternativas
  - Principal (políticas de seguridad): una entidad a la que se le puede otorgar acceso a objetos
  - Sujeto (sistemas operacionales que aplican una política): una entidad activa en un sistema IT

Sujetos, Objetos, Operaciones

- Objetos: archivos, memoria, impresoras, nodos en una red
- Sujeto: un proceso que se ejecuta bajo una cierta identidad
- Una simple distinción entre entidad pasiva y activa
- Dos opciones para especificar control, estableciendo
  - Lo que un sujeto puede hacer
  - Lo que se puede hacer con un objeto
- S (conjunto de sujetos), O (conjunto de objetos), A (operaciones de acceso)

Operaciones de Acceso

- Dependiendo del enfoque e interés, operaciones de acceso varían desde
  - lectura y escritura de archivos
  - a invocación de métodos en un sistema orientado a objetos
- Sistemas comparables pueden usar diferentes operaciones de acceso, y aún asociarle diferente significado a operaciones que aparentan ser las mismas

Modos de Acceso

- En el nivel más elemental
  - *Observe* : mirar el contenido de un objeto
  - *Alter* : cambiar el contenido de un objeto
- La mayoría de las políticas de control de acceso podrían ser expresadas en función de estas operaciones
- Formulación poco precisa y difícil de verificar

Tipos de Permisos en Bell-LaPadula

- Un nivel de complejidad superior:
  - *execute, append, read, write*
- *Relación entre tipos de permisos:*
  - *observe: read, write*
  - *alter: append, write*

Rationale

- Un archivo tiene que ser abierto (lectura, escritura) antes de que sea permitido el acceso. Acceso de escritura generalmente incluye acceso para lectura
- Pocos sistemas implementan la operación append: tiene sentido en audit logs
- SOs pueden utilizar archivos (ej. Programas) sin necesidad de abrirlos; execute no incluye ni observe ni alter

SO actuales

- Políticas de control de acceso son definidas en función de tres operaciones:
  - *read, write, execute*
- En Unix, acceso *write* no implica acceso *read*
- Aplicadas a directorios varía el significado:
  - read: lista el contenido del directorio
  - write: crea o renombra un archivo en el directorio
  - execute: buscar en el directorio

Propiedad de recursos

- Quién está a cargo de establecer las políticas de seguridad?
- Dos opciones fundamentales:
  - Se puede definir un propietario para cada recurso, y que sea el propietario quien decide quién puede acceder a ese recurso (**Discrecional**)
  - Una política que abarca todo el sistema establece los permisos de acceso (**Mandatoria**)
- La mayoría de los SO soportan el concepto de propietario de un recurso

Estructuras de Control de Acceso

- Definición de estructuras que permitan:
  - Establecer qué operaciones de acceso son permitidas
  - Respondiendo a los siguientes requerimientos
    - Que ayuden a expresar adecuadamente las políticas deseadas
    - Que permitan verificar que las mismas han sido correctamente formuladas

Matriz de Control de Acceso

- Permisos de acceso pueden ser establecidos individualmente para cada sujeto y objeto en términos de una *matriz de control de acceso*

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

Capabilities

- Los permisos son asociados a los sujetos
- Corresponde a la fila de permisos asociados a un sujeto en la MCA
- En el ejemplo
  - S_1: (edit.exe: execute; fun.com: execute, read)
  - S_2: (bill.doc: read, write; edit.exe:execute;fun.com: execute, read, write)
- Capabilities son asociadas con DAC

Capabilities

- Cuando un sujeto crea un nuevo objeto puede dar acceso a otro sujeto proporcionando las capabilities correspondientes
- No son un mecanismo ampliamente aceptado
  - Es dificil determinar dado un objeto quién tiene acceso al mismo
  - Es dificil revocar una capability
- Sistemas distribuidos han ayudado a reflotar el interés en estos mecanismos: politicas que tiene que considerar roaming de usuarios

Access Control Lists

- Una ACL asocia los permisos de acceso a un objeto con el objeto mismo
- Corresponde a una columna de la MA
- Característica de seguridad típica de SO comerciales
- En el ejemplo:
  - bill.doc: (S2: read, write)
  - edit.exe: (S1: execute; S2:execute)
  - fun.com: (S1: execute, read; S2: execute, read, write)

Access Control Lists

- Manejo de permisos basados en sujetos individualmente es complicado: grupos
- El modelo Unix de CA se basa en simples ACLs que asignan permisos a los principals *user*, *group*, *others*
- Difícil determinar los permisos de un usuario, por ejemplo, para revocarlos

Privilegios

- Centrando la atención en las operaciones de acceso, los privilegios son conjuntos de permisos para la ejecución de las mismas
- Típicamente, privilegios son asociados con funciones de SO y están relacionados con actividades de administración de sistemas, backup, acceso al mail o a la red.
- Privilegios pueden ser entendidos como una capa intermedia entre los sujetos y las operaciones

Seguridad Multinivel

- Investigación en el área en los 70 y 80 fue fuertemente motivada por demandas de protección de información clasificada
- Políticas existentes asignaban niveles de seguridad a documentos y clearances asociadas a usuarios determinaban a que documentos éstos podían acceder
- Las versiones mas elementales de políticas usaban una jerarquía linealmente ordenada de niveles: *unclassified, confidential, secret, top secret*

El Reticulado de Niveles de Seguridad

- Dada la política estándar de confidencialidad que establece que un sujeto puede observar a un objeto sólo si el nivel de seguridad del sujeto es mayor que el del objeto, se plantean las siguientes cuestiones:
  - Dados dos objetos con diferente niveles de seguridad, cuál es el mínimo nivel de seguridad que debe poseer un sujeto para poder observar los dos objetos?
  - Dados dos sujetos con diferente niveles de seguridad, cuál es el máximo nivel de seguridad que debe poseer un objeto para poder ser observado por los dos sujetos?

El Reticulado de Niveles de Seguridad

- La estructura algebraica que permite responder estas dos cuestiones es el *reticulado*
- Un reticulado (L,≤) consiste de un conjunto L y un orden parcial ≤ sobre L, tal que para todos dos elementos *a* y *b* de L existe un *least upper bound* (lub) *u* y un *greatest lower bound* (glb) *l*, tal que
  - a ≤ u, b ≤ u, y para todo v, si a ≤ v /\ b ≤ v => u ≤ v
  - l ≤ a, l ≤ b, y para todo k, si k ≤ a /\ k ≤ b => k ≤ l

Ordenes Parciales

- Un orden parcial (≤) sobre un conjunto (de etiquetas de seguridad) L es una relación binaria en L que es
  - reflexiva, transitiva y antisimétrica
- Ejemplos
  - El conjunto potencia de un conjunto X con la relación de inclusión
  - Los números naturales con la relación “divide a”

El Reticulado de Niveles de Seguridad

- En seguridad se dice que *a es dominado por b* si a ≤ b
  - El nivel de seguridad que es dominado por todo otro nivel se conoce como *System Low*
  - El nivel de seguridad que domina a todo otro nivel se conoce como *System High*
- Conocer la teoría de reticulados ayuda a entender mucho de los trabajos en el área de seguridad multinivel

Seguridad Multinivel

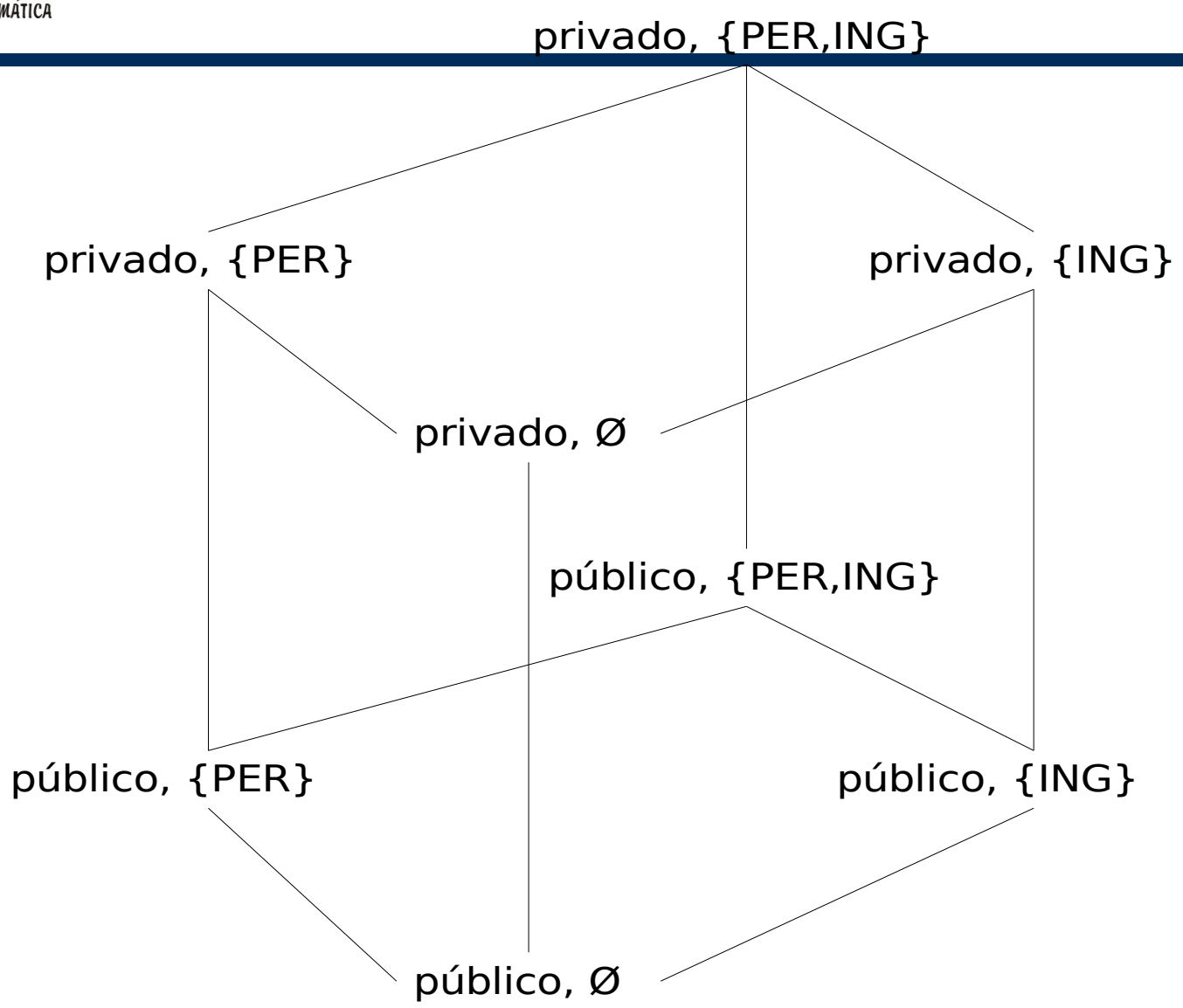
- Con un orden lineal de niveles sólo se pueden expresar políticas muy limitadas
- No es posible, por ejemplo, restringir el acceso a documentos de un proyecto secreto X sólo al personal trabajando en X: cualquiera a nivel *secret* podría tener acceso a los mismos
- Para poder definir políticas *need-to-know (least privilege)* que controlan el acceso a recursos de un proyecto específico se introdujo el siguiente reticulado de niveles de seguridad

Reticulado *need-to-know*

- Sea H un conjunto de *clasificaciones* con un orden lineal $\leq_H$
- Sea C un conjunto de *categorías* (nombre de proyecto, departamentos de una compañia, etc.). Un *compartimento* es un conjunto de categorías
- Una etiqueta de seguridad es un par (h,c) donde h es un nivel de seguridad y c un compartimento
- El orden parcial de etiquetas de seguridad se define como:
  - $(h_1,c_1) \leq (h_2,c_2)$ sii $h_1 \leq_H h_2$ y c_1 incluido en c_2

Ejemplo

- Dos niveles jerárquicos: público y privado
- Dos categorías: personal (PER), Ingeniería (ING)
- Orden:
  - (público,{PER}) ≤ (privado, {PER})
  - (público,{PER}) ≤ (privado, {PER, ING})
  - (público,{PER}) y (privado, {ING}) son incomparables


privado, {PER,ING}
privado, {PER}
privado, {ING}
privado, Ø
público, {PER,ING}
público, {PER}
público, {ING}
público, Ø

Bibliografía y Referencias

- **R. Anderson**, *Security Engineering – A Guide to Building Dependable Distributed Systems,* Wiley, 2001.
- **D. Gollman**, *Computer Security*, Wiley, 2006.
- **E. Bertino**, *Notes of Information Security course*, Purdue University, 2005.
- **R. Morris, K. Thompson**, *Password Security: A Case History*, Comm. ACM, vol. 22, 1979.
- **D. Klein**, *"Foiling the Cracker": A Survey of, and Improvements to, Password Security*, Proc. USENIX Security Workshop, 1990.

Bibliografía y Referencias

- **R.S. Sandhu**, *Lattice-Based Access Control Models*, IEEE Computer, 1993.
- **D. Denning**, *A Lattice Model of Secure Information Flow*, Comm. ACM, vol 19, 1976.