Tema 4: Protección y Seguridad

1. Introducción
2. Protección
3. Seguridad

1. Introducción

- **Protección**: es estrictamente un problema interno

  ¿Cómo proporcionamos un acceso controlado a los programas y datos almacenados en un sistema?

- **Seguridad**: considera el entorno externo dentro del cual opera el sistema

- Distinción entre políticas y mecanismos ⇒ flexibilidad
  - **Políticas (lo que se hará)**: qué datos serán protegidos
  - **Mecanismos (cómo se hará)**: Deben especificar que controles se van a realizar y qué medios van a hacer efectivos dichos controles
  - Las políticas varían, los mecanismos deben ser lo bastante generales para abarcar un amplio abanico de políticas

2. Protección

- Un sistema es una colección de procesos y objetos (hardware o software)
- Cada objeto tiene:
  - Un **nombre** único que lo identifica
  - Un conjunto de **operaciones** para acceder a él
  - **Ejemplos**: CPU - ejecutar
    semáforo - wait, signal
    archivo - leer, escribir, ejecutar, ...
- Problema de protección: asegurar que cada objeto es accedido correctamente y sólo por aquellos procesos que lo tienen permitido

Dominio de protección

- **Dominio de protección**: define un conjunto de objetos junto con sus derechos de acceso

  { <nombre_objeto>, <cjto_derechos_acceso> }

- **Derecho de acceso**: permiso para realizar una operación sobre un objeto
- Un proceso se ejecuta dentro de un dominio. La **asociación proceso-dominio** puede ser:
  - **Estática**: el conjunto de objetos disponibles para un proceso es fijo durante la vida del proceso
  - **Dinámica**: cambio de dominio durante la ejecución del proceso

Ejemplos de dominios de protección

Dominio 1

<F1, {leer}>

<F2, {leer, escribir}>

Dominio 2

<F3, {leer}>

<F4, {ejecutar}>

<Impresora, {escribir}>

<F1, {ejecutar}>

<F5, {leer, escribir, ejecutar}>

Dominio 3

- Los dominios no tienen porqué ser disjuntos. Ej. D2 y D3
- Un objeto se puede encontrar en más de un dominio con derechos distintos. Ej. F1 en D1 y en D3

Dominio de protección (y II)

- ¿Qué entidades pueden determinar un dominio?

  - Cada usuario: el conjunto de objetos accesibles depende de la identidad del usuario. Ejemplo, Unix

  - Cada proceso: el conjunto de objetos accesibles depende de la identidad del proceso

  - Cada procedimiento: el conjunto de objetos accesibles se corresponde con sus variables locales

- **Ejemplos**:

  - Modo dual de operación -> solo, insuficiente

  - Unix: los dominios se asocian con el uid. Los cambios de dominios se realizan a través del uid efectivo, son temporales y los proporciona el Sistema de Archivos

Matriz de acceso

- Es el modelo que nos proporciona un mecanismo apropiado para especificar distintas políticas, y definir e implementar un control estricto (tanto para asociación estática como dinámica)

- Elementos de la matriz de acceso:
  - Filas ↔ Dominios
  - Columnas ↔ Objetos
  - Celdas ↔ Derechos de acceso

Matriz de acceso (y II)

- Ejemplo de una matriz de acceso

Objeto / Dominio	F1	F2	F3	F4	F5	Impr.
D1	leer	leer escribir				
D2			leer	ejecutar		escribir
D3	ejecutar				leer escribir ejecutar	

Matriz de acceso (y III)

- Se pueden incluir los dominios como objetos para indicar el cambio entre dominios (conmutación)

Objeto / Dominio	F1	F2	F3	F5	Impres.	D1	D2	D3
D1	leer propiet.	leer escribir					Conmutar	
D2			leer*		escr			.
D3	eje.			leer escr. ejec.			Conmutar Control	

- Igualmente, cada entrada de la matriz puede incluirse como un objeto más a proteger

Implementación de la matriz de acceso

1. **Tabla global:** conjunto ordenado de triples <dominio, objeto, conjunto_derechos>

   Problemas:

   - Matriz de gran tamaño ⇒ no cabe en MP
   - Difícil agrupar dominios u objetos

2. **Listas de acceso para objetos (ACL):** almacena la matriz por columnas. Asocia a cada objeto una lista ordenada de pares <dominio, conjunto_derechos>

   - Ventaja: Fácil agrupar dominios
   - Problema: Saber para un dominio dado, su conjunto de <objeto, conjunto_derechos>

Implementación de la matriz de acceso (y II)

3. Listas de capacidades para dominios: almacena la matriz por filas. Asocia a cada dominio una lista ordenada de pares <objeto, conjunto_derechos>

- Cada elemento se denomina capacidad
- La listas de capacidades deben protegerse por el SO

4. Cerrojo-Llave:

- Cada objeto tiene una lista de patrones de bits únicos (cerrojos), y cada dominio otra lista (llaves)
- Un proceso en un dominio puede acceder a un objeto si tiene una llave que se corresponde con uno de los cerrojos del objeto

Implementación de la matriz de acceso (y III)

- La mayoría de sistemas utilizan una combinación de ACL y listas de capacidades. Ej. S. de archivos de Unix
- La ACL se corresponde con necesidades del usuario
- La lista de capacidades se utiliza para localizar información del proceso

- **Ejemplos**:

ACL (F1) = {<D1,leer>,
<D3,ejecutar>}

Lista de capacidades para D1:

	tipo	derechos	objeto
0	archivo	leer	nºi-nodo F1
1	archivo	leer/escr.	nºi-nodo F2
2	Capac.	conmutar	puntero a lista de capacidades de D2

Revocación de los derechos

- Es necesario revocar derechos a objetos compartidos en sistemas con un mecanismo dinámico de protección
- Distintos aspectos de la revocación:
  - Inmediata o retardada
  - Selectiva o general: ¿afecta a un usuario o a todos?
  - Parcial o total: ¿se eliminan todos los derechos?
  - Temporal o permanente: ¿disponibles de nuevo?
- Revocación con ACL: Es fácil e inmediata. Se busca en la lista los derechos a revocar y se eliminan
- Revocación con LC: Difícil. Las capacidades están distribuidas por todo el sistema

Revocación de los derechos (y II)

- Distintas implementaciones para la revocación en un sistema con capacidades:
  - **Readquisición**: Periódicamente se borran todas las capacidades
  - **Punteros**: El objeto tiene una lista de punteros para localizar sus capacidades. Para revocar sólo hay que seguir los punteros y modificar la capacidad. Costoso y general
  - **Indirección**: Las capacidades no apuntan directamente a los objetos, sino indirectamente. Imposible revocación selectiva
  - **Esquema de llaves**: Cada objeto tiene una llave maestra. Al crear la capacidad, se le asigna la llave. Revocar significaría cambiar la llave maestra

3. Seguridad

- Un sistema es seguro si sus recursos son utilizados y accedidos como se espera en todas las circunstancias
- Problema: intentar saltarse la protección de recursos
- Requisitos de seguridad: confidencialidad, integridad y disponibilidad de la información
- Las violaciones de seguridad pueden ser:
  - Accidentales: fáciles de tratar
  - Intencionadas:
    - lectura no autorizada a datos (viola la confidencialidad)
    - modificación no autorizada (viola la integridad)
    - destrucción (viola la disponibilidad)

Seguridad (y II)

- Se pueden tomar medidas de seguridad a 3 **niveles**:
  - Físico: el lugar donde se encuentra el sistema debe estar protegido físicamente
  - Humano: autorización cuidadosa a los usuarios
  - Sistema Operativo: identificación
- **Identificación**:
  - identifica programas, procesos y usuarios
  - ¿cómo? combinación de los siguientes conjuntos de elementos:
    - posesión de un objeto (tarjeta, llave,...)
    - atributo del usuario (huella dactilar, firma,...)
    - conocimiento del usuario (password, username)

Seguridad (y III)

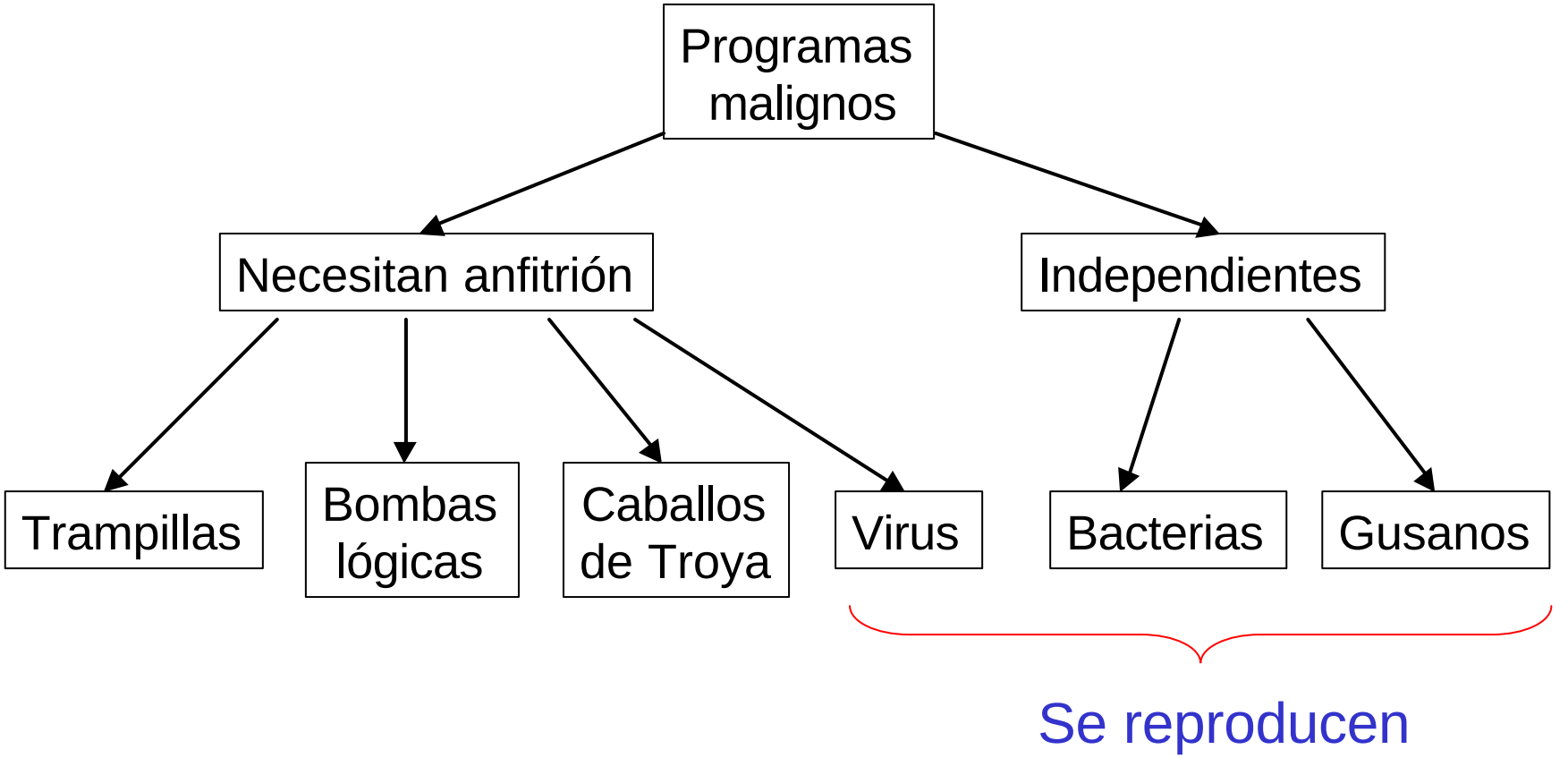
- Problema de los password: mantenerlos en secreto
- Los password pueden ser:
  - generados por el sistema: difíciles de recordar
  - seleccionados por el usuario: fáciles de adivinar
- Formas comunes de conocer un password:
  - conocer al usuario o tener información de él
  - intentar todas las posibles combinaciones de letras, números y otros caracteres (*crypt*)
- La lista de password de un sistema puede ser:
  - privada: acceso denegado a los usuarios
  - pública: se tiene acceso pero su contenido está codificado

Técnicas de intrusión

- Objetivo de los intrusos: obtener el acceso a un sistema o aumentar el conjunto de privilegios
- **Hacker**: talentos informáticos que entran al sistema por el placer de recopilar información que utilizan si la necesitan
- **Cracker**: acceden, y presumen de ello, a los sistemas para los que no tienen autorización. Suelen cometer daños irreversibles o irreparables en el sistema.
- **!Cuidado!** Ambos son ilegales

Técnicas de intrusión (y II)

- Clasificación general de las amenazas por software:



Técnicas de intrusión (y III)

- **Trampilla**: punto de entrada secreto e indocumentado dentro de un programa. Utilizado para conceder accesos sin necesidad de identificación (*Juegos de Guerrra*)
- **Bomba lógica**: lógica introducida en un programa que, cuando se dan ciertas condiciones en el sistema, ejecuta alguna función no autorizada
- **Caballo de Troya**: rutina secreta e indocumentada que se inserta dentro de un programa útil (ej. editor). La ejecución del programa origina la ejecución de la rutina

Técnicas de intrusión (y IV)

- **Virus**: código incrustado en un programa que hace que se inserte una copia de sí mismo en uno o más programas. Además, puede realizar una función no deseada
- **Bacteria**: programa que consume recursos del sistema mediante su reproducción
- **Gusano**: programa que puede reproducirse y enviar copias de sí mismo a través de la red. Además, puede realizar alguna función no deseada

Técnicas para mejorar la seguridad

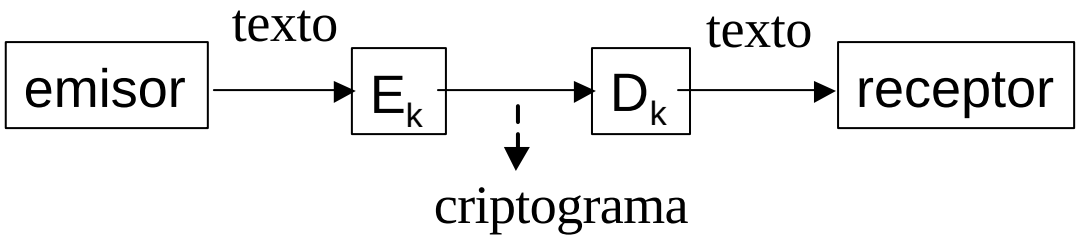
1. **Monitorización de amenazas**: comprobar si se intenta violar la seguridad del sistema ⇒ buscar patrones de actividad sospechosos
2. Revisión de la información (**auditoría**):
   - se registra el tiempo, usuario y tipo de todos los accesos a un objeto del sistema y si la operación tuvo o no éxito (archivo *log*)
   - una vez que la seguridad se ha violado, esta información sirve para:
     - determinar cómo y cuándo ocurrió el problema y el daño producido
     - recuperarse del ataque
     - desarrollar una seguridad mejor

Técnicas para mejorar la seguridad (y II)

3. **Explorar** periódicamente distintos aspectos del sistema para evitar los ataques:

- claves cortas o fáciles
- programas no autorizados en directorios del sistema
- procesos no esperados de larga duración
- inapropiada protección de directorios/archivos
- cambios de tamaño en programas del sistema
- programas con acceso para cambiar SUID no autorizados

4. Proteger los datos mediante **encriptación** (sobretodo de la información transmitida por la red, los passwords, etc.)



E: algoritmo encriptación
D: algoritmo desencriptación
K: llave o llaves para cada aplicación

Libro Naranja

- Desarrollado por el Ministerio de Defensa de EEUU
- Mide la fiabilidad y certifica que un sistema es seguro respecto a un conjunto de criterios de seguridad
- **Objetivos**:
  - proporcionar una métrica a los usuarios para valorar el grado de fiabilidad de un sistema
  - proporcionar a los diseñadores una directriz para construir productos comerciales fiables
  - proporcionar una base para indicar los requisitos de seguridad en las especificaciones de adquisiciones informáticas
- **Niveles** de seguridad: (**+**) A1 B3 B2 B1 C2 C1 D (**-**)

Conclusión

- El experto en seguridad, *Gene Spafford*, dijo:

  “El único sistema auténticamente seguro es el que está desconectado, desenchufado, y empaquetado en un recipiente hermético de titanio, guardado en un bunker de hormigón, rodeado de gas nervioso y custodiado por guardias muy bien armados y magníficamente pagados. E incluso así, yo no me jugaría la vida por él”

Conclusión: Un sistema es tan seguro como lo sea la gente que accede a él.