Sistemas Operativos

Seguridad en Sistemas Operativos

Curso 2023

Facultad de Ingeniería, URL

Agenda

1. Introducción
2. Motivación
3. Mecanismos de seguridad
   - Procesos
     - Modos de ejecución
     - Gestor de memoria
     - Sistema de archivos
     - Control de acceso
4. Control de acceso en UNIX

Introducción

Seguridad: definiciones

Seguridad

- La seguridad trata sobre la protección de activos
- Debemos conocer cuáles son los activos y su valor para nosotros
- Las medidas de protección se clasifican en:
  - Prevención
  - Detección
  - Reacción

Seguridad de la información

- Cuando el activo a proteger es la información,
- vamos a querer proteger la información de ataques a la:
  - confidencialidad
  - integridad
  - disponibilidad

Confidencialidad, Integridad y Disponibilidad

- Confidencialidad: prevención de difusión no autorizada de la información
- Integridad: prevención de modificación no autorizada de la información
- Disponibilidad: prevención de apropiación no autorizada de información o recursos

Seguridad Computacional

- *”Mecanismos necesarios para proveer protección y seguridad de un sistema computacional.”*
- *”Computer security deals with the prevention and detection of unauthorized actions by users of a computer system.”*
- Según esta definición, autorización y control de acceso son elementos claves
- *”Computer security is concerned with the measures we can take to deal with intentional actions by parties behaving in some unwelcome fashion.”*

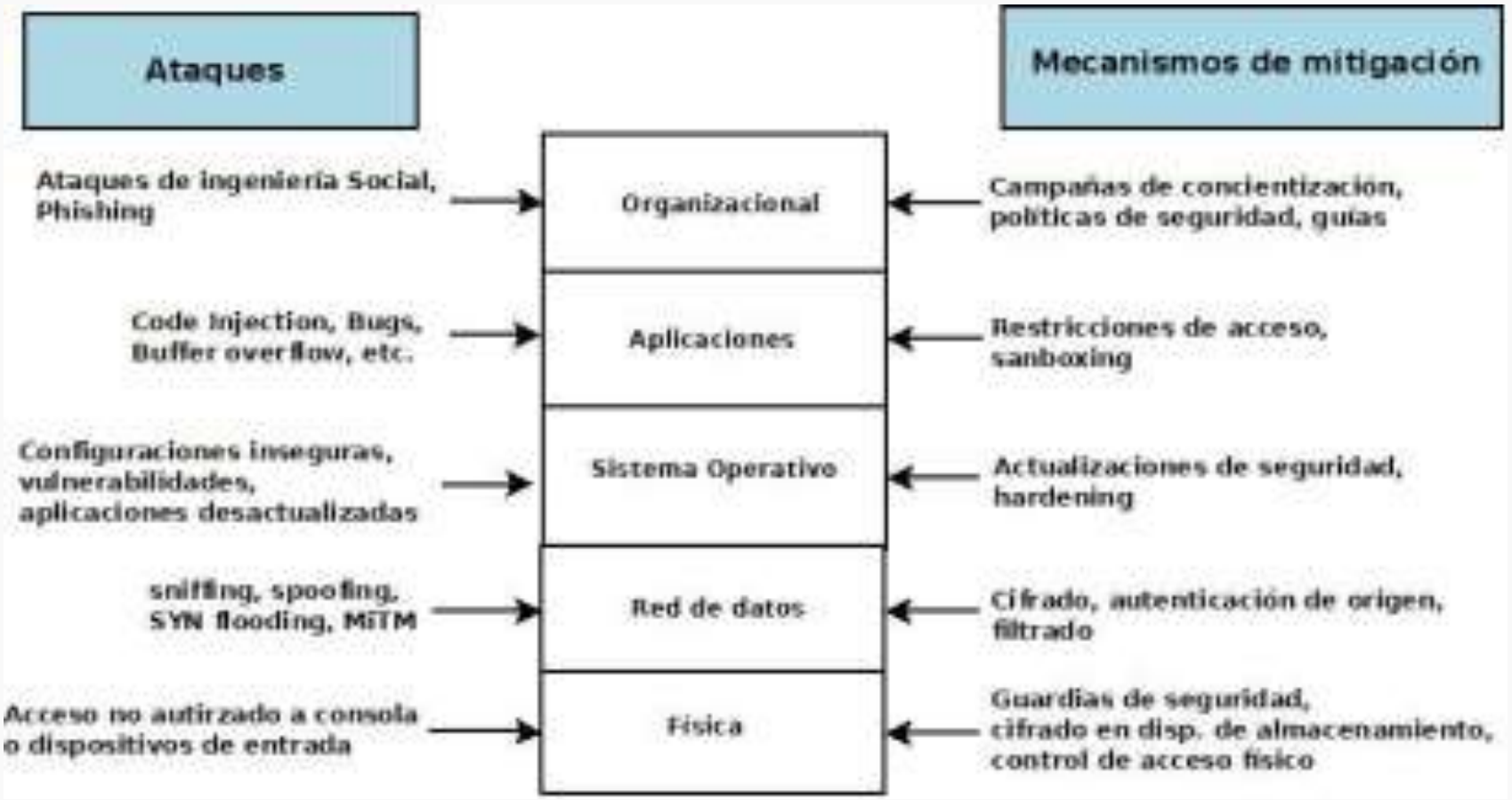
Principios de Seguridad

Lineamientos o principios metodológicos que sirven de guía para implementar controles de seguridad

- Asegurar el eslabón (punto) más débil
- Defensa en profundidad
- Principio de menor privilegio
- Reducir la superficie de ataque
- Compartimentar (sandboxing)
- Asegurar los valores por defecto
- Fallar en forma segura
- Security Through Obscurity

Principios de Seguridad

Debemos aplicarlo con una visión integral



Motivación

¿Por qué es necesario?

Intermediario entre el hardware y los usuarios del sistema, encargado de:

- asignación eficiente de recursos entre los procesos y usuarios del sistema
- proteger el sistema de programas (y usuarios) maliciosos
- mantener y proteger espacios de usuarios disjuntos
- permitir a los usuarios el acceso a datos, programas y otros recursos
- ejecutar programas de usuarios
- optimizar la eficiencia total del sistema
- gestionar dispositivos de entrada y salida

Desde que el sistema se inicia hasta que es apagado

¿Por qué es necesario?

Recursos:

- CPU (Gestor de procesos)
- Memoria virtual o de swap (Gestor de memoria)
- Almacenamiento secundario en disco (File System)
- Terminales, impresoras, dispositivos de I/O
- Acceso a la red, ancho de banda (Networking)

Todos ellos pueden ser mal utilizados o dañados intencional–mente o no.

Mecanismos de seguridad

¿Qué podemos hacer a nivel de los S.O.?

Controlar:

- qué personas (usuarios) pueden utilizar el sistema
- qué programas un usuario puede ejecutar (procesos)
- los recursos que los procesos pueden acceder

Proteger:

- procesos entre si, que comparten recursos del sistema
- integridad del sistema operativo (todo?)

Auditar

- La actividad de los usuarios (procesos), así como
- cumplimiento (validez) de los controles

Mecanismos de seguridad

¿Qué mecanismos ofrecen?

- Procesos
- Modos de ejecución (Kernel vs User Mode)
- Sistema de memoria virtual
- Sistema de gestión de archivos (File Systems)
- Identificación y Autenticación (IA)
- Control de acceso y autorización
- Registrar la actividad (auditoría)
- Virtualización / Sandboxing

Procesos

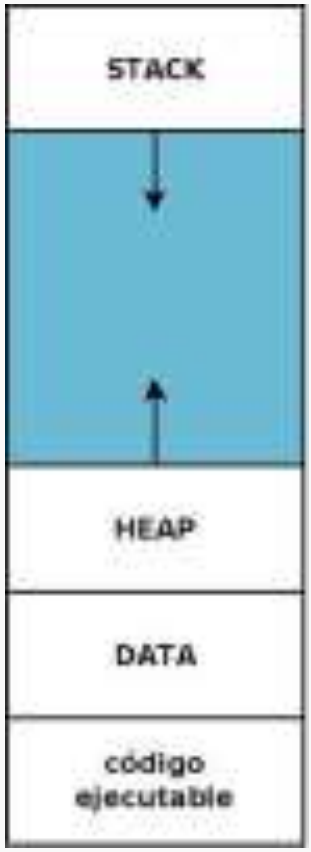
- Son la abstracción central en el sistema operativo
- Es la forma en que que los usuarios interactúan en el sistema
- Son independientes, y no deberían influenciarse directamente unos a otros
- El control de acceso es organizado en torno a los procesos = programas en ejecución
- Se identifican con un identificador único de proceso (Process ID o PID)

Procesos

El sistema operativo mantiene la siguiente información:

- Código (del programa) que está ejecutando
- Memoria asignada
- Valor del *program counter*
- Valor de los registros
- Stack de ejecución
- Otros recursos asignados como p.e. archivos abiertos

Proceso en memoria



Código vulnerable

```
#include <stdio.h>
int main(int argc, char **argv)
{
    char buf[8]; // buffer for eight characters
    gets(buf); // read from stdio (sensitive function!)
    printf("%s\n", buf); // print out data stored in buf
    return 0; // 0 as return value
}
```

Procesos: problemas de seguridad

Malware:
Programa que es insertado en forma encubierta dentro de otro programa con el objetivo de destruir datos, ejecutar programas destructivos o intrusivos; o que comprometan la confidencialidad, integridad o disponibilidad de los datos, aplicaciones o el propio sistema operativo de la victima. (Def. según NIST SP 800-83r1)

Formas de malware

- Virus
- Worms (gusanos)
- Ransomware

Modos de ejecución: User vs Kernel mode

- La mayoría de los S.O. distinguen diferentes modos de ejecución o niveles de privilegio
  - Kernel, supervisor o system mode
  - User mode
- El modo kernel ofrece privilegios sobre las funciones del sistema
  - acceso completo a toda la memoria,
  - todas las instrucciones soportadas por el CPU
  - y demás recursos gestionados por el S.O.
- User mode ofrece un ámbito de ejecución para procesos no críticos
- En general estos modos cuentan con soporte de hardware (flags a nivel de registros de ejecución del procesador)

Modos de ejecución: Invocaciones controladas

- Se producen cuando un proceso en modo usuario requiere ejecutar una operación que requiere modo supervisor
- debido al riesgo de estas operaciones, es necesario:
  - que el sistema ejecute solo un conjunto de operaciones predefinidas (limitadas y bien definidas) y una vez finalizada
  - controlar que el proceso vuelva a User Mode antes de devolver el control al usuario

Modos de Invocación: objetivos de seguridad

- El kernel del S.O. ejecuta en modo kernel
- Es importante proteger:
  - integridad de los procesos ejecutando en modo kernel
  - al sistema operativo de malware

Gestor de memoria

Es responsable de:

- asignar memoria a los procesos (in a fair way)
- el manejo de memoria virtual en sus distintas formas:
  - segmentos de largo fijo (paginado),
  - largo variable
  - o swapping

Debe garantizar protección de memoria, esto es:

- Procesos no deberían poder acceder a la memoria de otros procesos o del propio S.O.

Gestor de memoria: problemas de seguridad

- Los mecanismos de protección no previenen que un proceso acceda a la memoria de otro luego que este último la libera
- Usuarios o procesos maliciosos pueden buscar datos sensibles en memoria recién asignada a estos
- el disco duro puede contener páginas de memoria de procesos, incluso después de apagado el computador (*swap*)

Sistema de archivos

- Mecanismo de abstracción sobre dispositivos de almacenamiento persistente
- Puede usarse como capa de abstracción de otros dispositivos de I/O (teclado, pantalla, interfaz de red, etc)
- La persistencia es una característica
  - buena para implementar disponibilidad
  - mala para implementar confidencialidad
- Capa de abstracción usada como base para el control de acceso

  quien tiene los permisos para acceder a cuales archivos o directorios

Sistema de archivos: problemas de seguridad

- Los mecanismos de control de acceso y seguridad que provee el S. O. pueden ser anulados retirando el disco duro

Contramedida: cifrar disco duro

- debemos tener cuidado con el manejo de claves
- Criptografía puede resolver algunos problemas de seguridad, pero siempre introduce uno nuevo: el manejo de claves

Sistema de archivos: problemas de seguridad

- El borrado en la mayoría de los File Systems de S.O. no hace un borrado físico!
  - Journaling, registros de transacciones, papeleras de reciclaje
  - Caches, swap o archivos de paginado
  - Datos viejos que quedan en bloques del File System en disco
  - Sectores del disco sin asignar o marcados como dañados o defectuosos
- Necesitamos políticas para el manejo de dispositivos viejos o defectuosos

Control de acceso y Autorización

- Establecer los permisos de acceso (“access rights”) es equivalente a especificar:

Quien puede hacer que sobre (quien o que)

- Hacerlos cumplir (enforcement)

Hacer chequeos antes de cualquier operación

- Auditar

Registrar y chequear los logs o registros de las acciones

Concepto aplicable no solo a S.O. sino que también a aplicaciones

Control de acceso: definiciones

- Seguridad ≡ Regular el acceso a los activos del sistema
- Identificación: forma en que una persona (principal) se identifica ante un sistema (*username*)
- Autenticación: proceso de verificación de la identidad (pretendida) de un usuario (principal) ante un sistema a través de algo que se sabe, se tiene o se es
- Objeto: entidad pasiva que requiere acceso controlado (por ejemplo archivos, impresora, etc.)
- Operaciones de acceso: formas de acceder a los objetos

Control de acceso: definiciones

- Usuarios inician sesiones/sujetos en computadoras que van a acceder a objetos a través de operaciones de acceso
- Sesiones con los mismos permisos son agrupados en dominios de protección
- La política de un sistema nos dice que permisos tiene cada dominio de protección
- Llamamos ”reference monitor” a la entidad que hace que se cumpla (*enforce*) la política de seguridad en cada intento de acceso a los objetos

Control de acceso y dominios de protección

Control de acceso: DAC vs MAC

Discretionary Access Control (DAC)
Si un usuario individual puede gestionar el mecanismo de control de acceso para permitir o denegar acceso a un recurso, ese mecanismo constituye un control de acceso discrecional (DAC), también llamado control de acceso basado en identidad (IBAC)

Mandatory Access Control (MAC)
Cuando un mecanismo de un sistema controla accesos a un objeto y un individuo (principal o sujeto) no puede alterar ese acceso, entonces el control de acceso es mandatorio (MAC) (ocasionalmente llamado control de acceso basado en reglas)

Control de acceso en UNIX

Modelo de seguridad en UNIX

- Los controles de seguridad en UNIX no están contemplados en los objetivos de diseño iniciales (1970)
- Nuevos controles de seguridad se han incorporado, otros se fortalecieron a demanda
- Siempre buscando interferir lo menos posible con las estructuras existentes
- La seguridad es gestionada por administradores con mucha experiencia

Principals

- En Unix son identificados por la pareja user identity (UID) y group identity (GID)
- Ambos enteros de 16 bits
- Algunos UIDs tienen significados especiales
- El super usuario en unix es siempre el uid 0

Grupos Unix

- Cada usuario está presente en al menos un grupo de usuarios (grupo primario)
- Agregar usuarios a grupos es una base conveniente para decisiones de control de acceso
- Ejecutando el comando *groups* se pueden saber los grupos a los que pertenece un usuario

Sujetos

- Los sujetos son los procesos del sistema
- Se identifican con un Process ID (PID)
- Se crean nuevos mediante fork o exec
- Cada proceso tiene asociado un UID/GID real, y uno efectivo
- El UID real (ruid) es heredado del padre
- El UID efectivo (euid) es heredado del padre o del archivo que se está ejecutando

Proceso de login

- Los usuarios se identifican mediante nombres de usuario y passwords
- Cuando un usuario se loguea, el proceso *login* verifica el usuario y password
- Si la verificación es exitosa, se cambia el UID/GID al del usuario y se ejecuta la *shell de login*
- Los passwords se cambian mediante el comando passwd

Objetos

- En UNIX, los objetos para el control de acceso incluyen: archivos, directorios, dispositivos, I/O, etc.

todo objeto es un archivo

- Están organizados en un sistema de archivos con estructura de árbol

Inodos

- Cada entrada de archivo en un directorio es un puntero a una estructura de datos llamada *inodo*
- Cada directorio tiene un puntero a sí mismo, el archivo '.',y un puntero a su padre '..'
- Cada archivo tiene un usuario dueño, usualmente el que lo creó, y un grupo dueño al que pertenece

Permisos de archivos

- Los permisos de los archivos (bits de permisos), se agrupan en tres tripletas
- Definen permisos de *read*, *write*, y *execute*, para el propietario (u), el grupo (g), y otros (other) o el resto del mundo (o)
- se pueden representar como números decimales, separando los nueve permisos (bits) en 3 grupos de 3 (ver ejemplos)
- Cada derecho de acceso está representado por un bit, el cual si está “prendido” permite el acceso

Permisos de archivos: ejemplos

Permisos UNIX

Permisos por defecto

- Las archivos son creados por defecto con permisos 666 y los directorios *777*
- Deben ajustarse con la *umask*
- Los permisos se calculan a partir del AND binario de los valores por defecto y el inverso de la máscara (NOT máscara) ≡ (XOR máscara)
- Es modificado con el comando *umask*

Ejemplos de UMASK

- umask 022: provoca que solo el propietario pueda modificar los archivos
- umask 027: deja sin permisos de escritura al grupo y ningún permiso al resto del mundo

Permisos para directorios

- Cada usuario tiene un directorio home (*Home Dir*)
- Se crean con el comando *mkdir*
- Para agregar archivos en un directorio, se deben tener los permisos adecuados
    - Permisos de lectura permiten a un usuario encontrar (listar) archivos en el directorio
    - Permisos de escritura permiten a un usuario agregar y borrar archivos al directorio
    - Permisos de ejecución se requieren para hacer el directorio el actual y para abrir archivos

Algoritmo de control de acceso

- Está basado en atributos de los sujetos (procesos) y de los objetos (recursos)
- Las sistemas Unix clásicos asocian tres conjuntos de derechos de acceso a cada recurso, correspondientes al owner (u), group (g) y world o el resto del mundo (o)
- El super usuario no está sujeto a estos chequeos

Algoritmo de control de acceso

- Si el uid efectivo (euid) del proceso indica que es dueño del archivo, los bits de permisos de owner deciden si se tiene acceso
- Si no se es dueño del archivo, pero el gid efectivo (egid) indica que su grupo es el dueño, se aplican los permisos del grupo
- Si no, se aplican los permisos de resto del mundo (Others)
- Se puede tener un archivo donde los permisos del dueño tenga menos permisos que el resto del mundo (others) !!

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez, sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez, sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez, sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos del grupo dueño

Ejemplos control acceso UNIX

Ejemplo 1

- El proceso con UID/GID efectivos *jose.rodriguez, sistoper*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos del grupo dueño
- El acceso solicitado será denegado

Ejemplos control acceso UNIX

Ejemplo 2

- Si en cambio, el proceso cuenta con UID/GID efectivos *jose.rodriguez, arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Ejemplos control acceso UNIX

Ejemplo 2

- Si en cambio, el proceso cuenta con UID/GID efectivos *jose.rodriguez, arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

Ejemplos control acceso UNIX

Ejemplo 2

- Si en cambio, el proceso cuenta con UID/GID efectivos *jose.rodriguez, arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos de others

Ejemplos control acceso UNIX

Ejemplo 2

- Si en cambio, el proceso cuenta con UID/GID efectivos *jose.rodriguez, arqsis*
- desea permisos de *lectura (r)* sobre
- el archivo ejemplo.txt del cual
  - *pedro.martinez, sistoper* es el UID, GID dueño y
  - *rw-,-wx,rwx (637)* son los permisos definidos

Resultado

- Se chequean los permisos de others
- El acceso solicitado será otorgado

Control de acceso en UNIX: limitaciones

- Los objetos en Unix solo tienen un usuario y un grupo dueño
- Los permisos solo controlan el acceso de lectura, escritura y ejecución
- Otras operaciones de control de acceso deben implementarse a nivel de aplicaciones
- Se hace impracticable implementar políticas de seguridad más complejas

Control de acceso en UNIX: limitaciones

- Estas limitaciones han llevado a que se hayan incorporado distintas “extensiones” o mejoras al modelo de control de acceso de Unix tradicional
- Algunos ejemplos:
  - Access Control Lists (ACLs) a nivel de archivos
  - Variantes del kernel de linux “seguras”: SeLinux, Trusted Solaris, etc
  - Mejoras en capas intermedias como p.e. Sudo, TCP wrappers

Set User ID (SUID)

- En ciertas circunstancias necesitamos poder ejecutar procesos con privilegios de root
  - abrir un socket en puertos < 1024 (privilegiado)
  - o modificar la contraseña de un usuario
- En sistemas UNIX, la solución adoptada es el uso de Set UserID (SUID) o Set GroupID (SGID)
- Programas con el SUID o SGID ejecutan con el UID/GID efectivo del usuario o grupo dueño del archivo

SUID root

- En archivos con SUID root, el usuario que lo ejecuta obtiene privilegios de root durante la ejecución
- Algunos ejemplos:
  - */bin/passwd*: cambio de contraseña
  - */bin/login*: programa de login
  - */bin/su*: Cambiar el UID (Switch Userid)
- Debemos cuidar que estos programas hagan solo lo que deben hacer

Riesgos de SUID root

- Si logramos *"engañar"* a un programa con SUID de root, estamos logrando acceso de root
- Estos programas deben procesar con mucho cuidado los parámetros de entrada :-)
- Usemos SUID/SGID sólo cuando es estrictamente necesario
- Debemos controlar en especial la integridad de estos programas (Tripwire, AIDE, Yafic)

Bibliografía

- Dieter Gollmann, Computer Security, 3rd edition, Editorial Wiley, 2011. Cap. 3, 6 y 7.
- A. Silberschatz, P. Galvin and G. Gagne, Operating Systems Concepts, 9th edition, Editorial Wiley, 2014. Cap 14 y 15.
- M. Souppay, K. Scarfone, Guide to Malware Incident Prevention and Handling for Desktop and Laptops, NIST Special Publication SP 800-53. Cap. 2