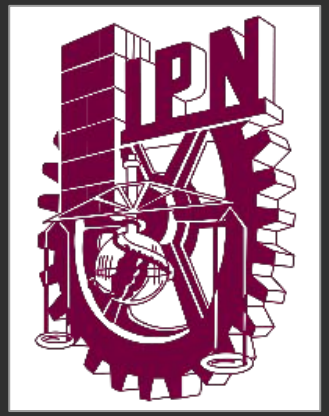


INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS OPERATIVOS

Unidad 6
Actividad 6

INTEGRANTES DEL EQUIPO:

COLIN RAMIRO JOEL
HERNÁNDEZ REYES JULIO CÉSAR
MALDONADO CERÓN CARLOS
MENDOZA GARCÍA ELIÚ EDUARDO

GRUPO: 4CM1

PROFESOR: CORTÉS GALICIA JORGE

1. Seguridad

1.1. Problema de la Seguridad

Podemos decir que un sistema es seguro si sus recursos se utilizan de la forma prevista y si se accede a ellos tal como se pretendía, en todas las circunstancias.

- Desafortunadamente no es posible conseguir una seguridad total.
- Las violaciones de seguridad de un sistema se clasifican en 2 categorías.
  a. Intencionadas (malintencionadas)
  b. Accidentales

Una amenza es la posibilidad de que exista una violación de seguridad por ejemplo el descubrimiento de una vulnerabilidad.

Un ataque es un intento de romper la seguridad.

- **Ruptura de la Confidencialidad.-** Implica la lectura no autorizada de ciertos datos o el robo de información. Es el objetivo de los intrusos.
- **Ruptura de la Integridad.-** Implica la modificación no autorizada de los datos. Pueden provocar que se atribuya una cierta responsabilidad a alguien inocente o que se modifique el código fuente de alguna aplicación comercial importante.
- **Ruptura de la Disponibilidad.-** Implica la destrucción no autorizada de datos
- **Robo de Servicio.-** Implica el uso no autorizado de recursos. Por ejemplo, un intruso puede instalar un demonio en un sistema que actúe como servidor de archivos.
- **Denegación de Servicio.-** Implica impedir el uso legítimo del sistema. Los ataques de denegación de servicio (DOS, denial-of-service) son accidentales.

1.2. Amenazas Relacionadas con los Programas

1.2.1. Caballo de Troya

- Es un segmento de código que utiliza inapropiadamente su entorno.
- Las rutas de búsqueda de gran longitud, agravan este problema.
- Una variante de esta amenaza, es un programa que emula el programa de inicio de sesión
- Otra variante es el **spyware.**

- Un usuario que no sea advertido, intentará iniciar la
  sesión en una terminal y observará que aparentemente
  ha escrito mal su contraseña, después
  vuelve a intentarlo y esta vez con éxito.

1.2.2. Puerta Trasera

- Se ilustra en la película Juegos de Guerra
- Ejemplo: El código puede tratar de detectar un ID de usuario o una contraseña específicos y al detectarlo, evitar los procedimientos de seguridad normales
- Se han dado casos de programadores, que han sido condenados por estafar a los bancos incluyendo errores de redondeo en su código y haciendo que esas fracciones de céntimo fueran abonadas en sus cuentas.
- Puede incluirse una puerta trasera inteligente dentro del propio compilador.

- Las puertas traseras plantean un difícil problema, debido a que para detectarlas, se tiene que analizar todo el código fuente de todos los componentes de un sistema.

1.2.3. Bomba Lógica

- Se trata de un programa capaz de iniciar un incidente de seguridad sólo cuando se dan determinadas circunstancias.
- Es difícil de detectar porque, en condiciones normales de operación, no existe ningún agujero de seguridad.
- Cuando se satisficiera un conjunto predefinido de parámetros, se crea el agujero de seguridad.

1.2.4. Desbordamiento de pila y búfer

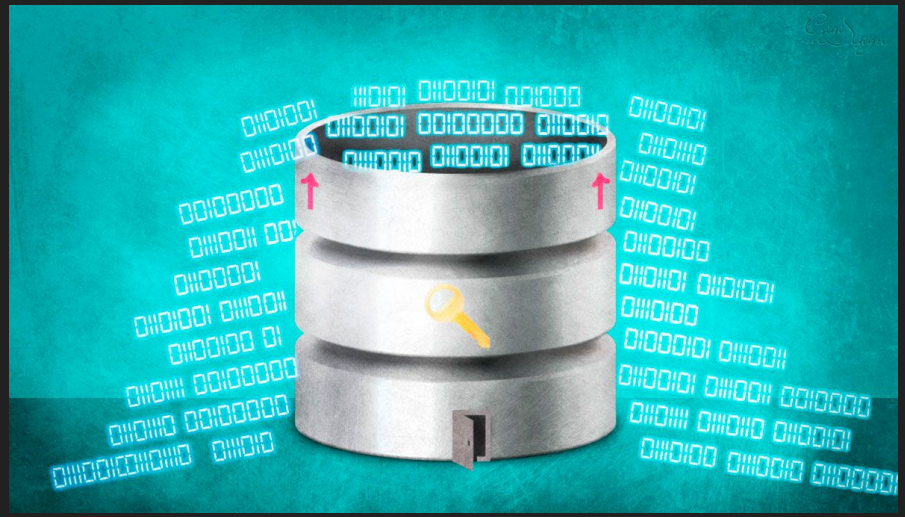
- Es la forma más común para que un atacante externo al sistema, a través de una conexión de red o de acceso telefónico, obtenga acceso no autorizado al sistema objetivo.
- Los usuarios autorizados del sistema también pueden utilizar este tipo de ataque para escalar sus privilegios.
- Lo que hace el programa es aprovechar un error.

```
#include <stdio.h>
#define BUFFER_SIZE 256

int main(int argc, char *argv[])
{
    char buffer[BUFFER_SIZE];

    if (argc < 2)
        return -1;
    else {
        strcpy(buffer,argv[1]);
        return 0;
    }
}
```

Programa en C con una condición de desbordamiento de búfer



1.2.5. Virus

- Son auto-replicantes y están diseñados para “infectar” otros programas.
- Pueden causar estragos en un sistema modificando o destruyendo archivos y provocando funcionamientos inadecuados de los programas y fallos catastróficos del sistema.
- Se puede ver también como el fragmento de código integrado dentro de un programa legítimo.
- Son muy específicos de las arquitecturas, de los S.O y de las aplicaciones.

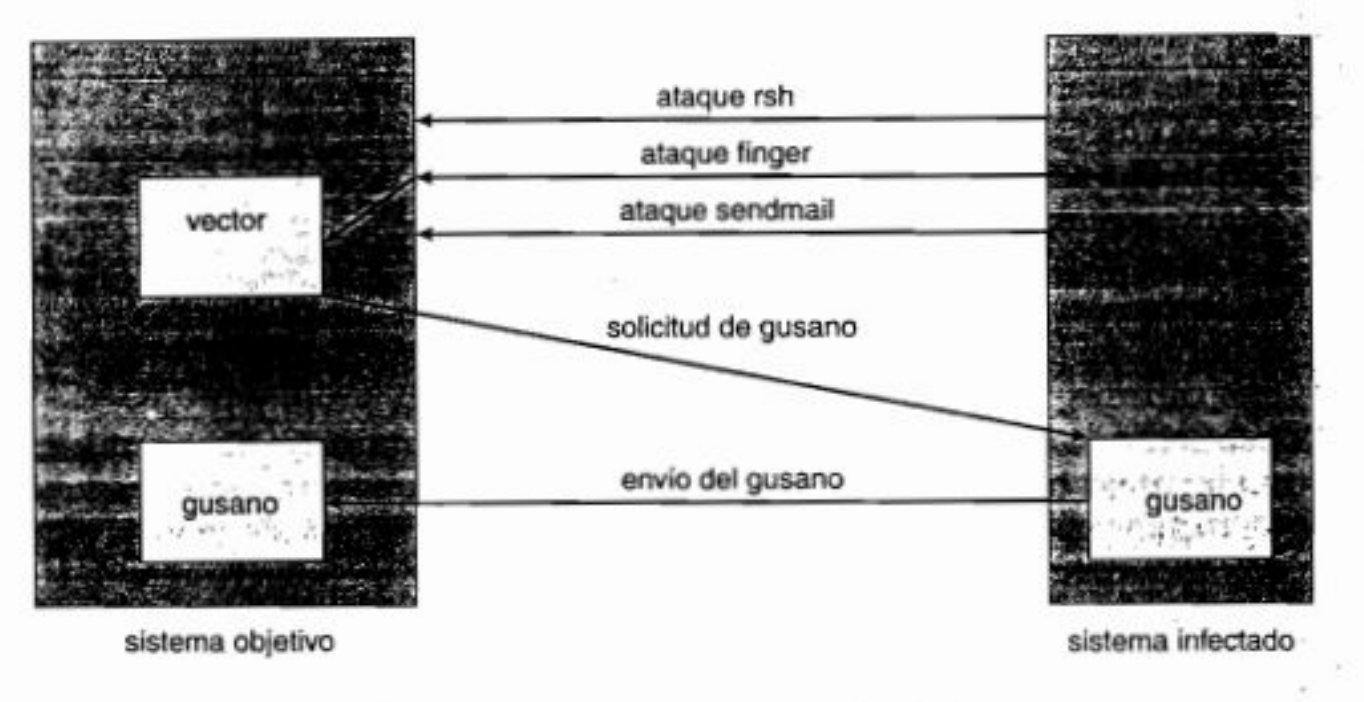
- Constituyen un problema especialmente grave para los usuarios de PC
- Suelen Propagarse a través de correo electrónico, siendo el correo basura el vector más común.
- También cuando los usuarios descargan programas infectados de servicios de comparación de archivos.

1.3. Amenazas del Sistema y de la Red

- Implican el abuso de los servicios y de las conexiones de red.
- Se utiliza para lanzar un ataque de programa
- Crean una situación en la que se utilizan inapropiadamente los recursos del S.O y los archivos de usuario.

1.3.1. Gusanos

★ Es un proceso que utiliza un mecanismo de reproducción para afectar al rendimiento del sistema.
★ Crea copias de sí mismo, utilizando recursos del sistema y en ocasiones impidiendo operar a todos los demás procesos.
★ Son potentes, debido a su capacidad de reproducción.
★ Además de colapsar una red completa.



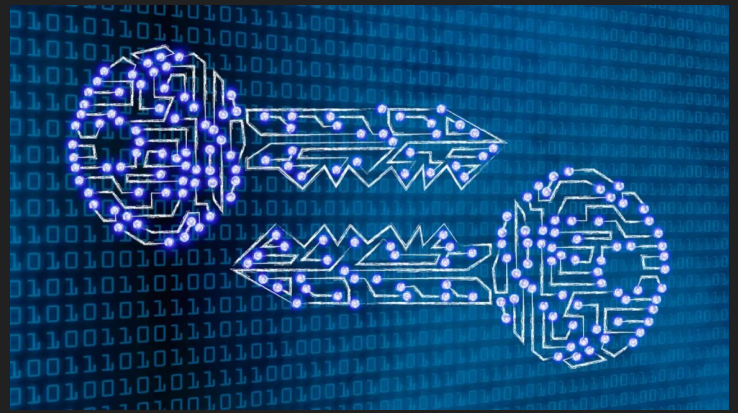
El Gusano Internet de Morris

1.4. La Criptografía como Herramienta de Seguridad

La Criptografía es la herramienta de carácter general a disposición de los usuarios y de los diseñadores de sistemas.

1.4.1. Cifrado

- Es un medio de restringir los posibles receptores de un mensaje.
- Un algoritmo de cifrado permite al emisor de un mensaje garantizar que sólo pueda leer el mensaje una computadora la cual posea una cierta clave. Dicho Algoritmo consta de los siguientes componentes:
    - Conjunto K de Claves
    - Conjunto M de Mensajes
    - Conjunto C de Mensajes de Texto Cifrado



Cifrado Simétrico

- *Se utiliza la misma clave para cifrar y para descifrar.*
- *El más comúnmente utilizado es el DES(data-encryption standard).*
- *El DES, funciona tomando un valor de 64 bits, una clave de 56 bits y realizando una serie de transformaciones.*

Cifrado Asimétrico

- *Las claves de cifrado y descifrado son distintas*
- *El más conocido es el RSA (Rivest, Shamir y Adleman).*
- *Se trata de un algoritmo de cifrado de bloque de clave pública.*
- *Sin embargo, los algoritmos asimétricos basados en curvas elípticas están ganando cada vez más terreno, ya que la longitud de clave de dichos algoritmos puede ser más corta para un grupo determinado de fortaleza criptográfica.*

1.5. Autenticación de Usuario

Si un sistema no puede autenticar a un usuario, entonces autenticar un mensaje procedente de dicho usuario no sirve de nada. Es por eso que este es un problema de seguridad importante para el S.O.

1.5.1. Contraseñas

- Se trata del método más habitual para autenticar la identidad de un usuario.
- A menudo, en ausencia de esquemas de protección más completos, se utilizan contraseñas para proteger objetos del S.O.
- Pueden considerarse un caso especial de claves

1.5.2. Vulnerabilidad de las Contraseñas

- Las contraseñas son extremadamente comunes, debido a que son fáciles de comprender y utilizar. Es por eso que es muy común que se adivinen, se muestren por accidente, sean interceptadas ó transferidas ilegalmente.
- Además de eso, pueden ser averiguadas mediante mecanismos de monitorización visual o electrónica.
- Alternativamente cualquier persona con acceso a la red en la que reside la computadora, puede añadir sin problemas un monitor de red, que le permita ver todos los datos ingresados por otro usuario(Sniffing).

2. Protección

Los procesos en un S.O. se deben proteger de las actividades realizadas por otros procesos, para proporcionar dicha protección, se pueden implementar diversos mecanismos para garantizar que **ÚNICAMENTE** los procesos que hayan obtenido la adecuada autorización del S.O. puedan operar sobre los archivos, segmentos de memoria, sobre la CPU y otros recursos del sistema.

2.1. Objetivos de la Protección

- Conforme han evolucionado los sistemas informáticos, su rango de aplicaciones se ha ido incrementado, con esto, también creció la necesidad de proteger la integridad de esos sistemas.
- Se necesita proporcionar protección por diversas razones:
    - Impedir una violación maliciosa e intencionada de una restricción de acceso por parte de un usuario.
    - Garantizar que cada componente de programa activo en un sistema utilice los recursos del sistema sólo en formas coherentes.

- El papel de la protección en un S.O. es proporcionar un mecanismo para la imposición de las políticas que gobiernen el uso de los recursos necesarios

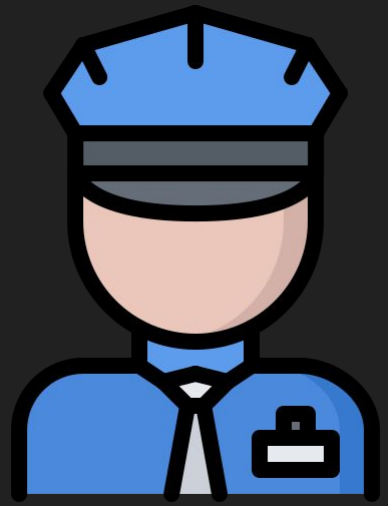
2.2. Principios de la Protección

Uno de los principios que ha permanecido por mucho tiempo a la hora de proporcionar protección es **EL PRINCIPIO DEL MÍNIMO PRIVILEGIO.**

- Dicta que a los programas, usuarios e incluso a los sistemas, se les conceda únicamente los privilegios suficientes para llevar a cabo sus respectivas tareas.

  Ejem:

  ★ Considere un Guardia de seguridad con una tarjeta magnética.
  ★ Si dicha tarjeta le permite el acceso únicamente en las áreas públicas que esté vigilando, un mal uso de esta, provocará un daño.
  ★ En cambio, si la tarjeta, le permite acceder a todas las áreas, el daño derivado de cualquier situación, será mucho peor.



2.3. Dominio de Protección

Un Sistema Informático se trata de una colección de procesos y objetos.

- Un objeto se define como el hardware(CPU, segmentos de memoria, impresoras, discos, etc) y el software(archivos, programas, semáforos, etc).
- Son tipos abstractos de datos.
- Cada objeto tiene un nombre distinto que lo diferencía de todos los demás objetos del sistema y sólo se puede acceder a cada objeto mediante operaciones bien definidas.
- Las operaciones dependen de cada objeto.

- ❏ Ahora bien, a un proceso sólo se le debe permitir acceder a aquellos recursos para los que tenga autorización.
- ❏ Un proceso sólo debería poder acceder a aquellos recursos que necesite actualmente para completar su tarea.

2.3.1. Estructura de Dominios

- Un dominio es una colección de derechos de acceso, cada uno de los cuales se trata de una pareja ordenada **<nombreObjeto, conjuntoDerechos>**
- Un proceso opera dentro de un **DOMINIO DE PROTECCIÓN**, el cual especifica a los recursos a los que este proceso puede acceder.
- Cada dominio define un conjunto de objetos y los tipos de operaciones que pueden invocarse sobre cada objeto.
- La capacidad de ejecutar una operación sobre un objeto, se le llama **DERECHO DE ACCESO**
- La asociación entre un proceso y un dominio puede ser **estática**, si el conjunto de recursos disponibles para el proceso fijo durante la vida del proceso, o bien sería **dinámica.**
- Los dominios dinámicos de protección es más complicado que establecer dominios estáticos de protección.

2.4. Matriz de Acceso

El modelo de protección puede contemplarse de forma abstracta como una matriz, llamada **MATRIZ DE ACCESO.**

- Sus filas representan dominios y las columnas objetos.
- Cada entrada de la matriz está compuesta de un conjunto de derechos de acceso.
- La entrada ***access(i,j)*** define el conjunto de operaciones que un proceso que se ejecute en el dominio Di puede invocar sobre el objeto Oj.

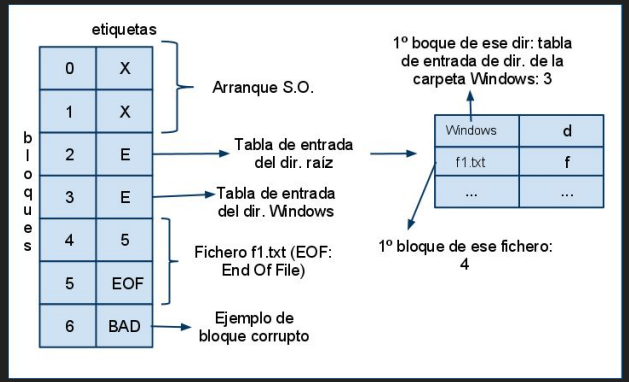
Dominio \ Objeto	A1	A2	A3	COM1	LPT1
D1	Leer		Leer		
D2				Leer	Imprimir
D3		Leer	Ejecutar		
D4	Leer Escribir		Leer Escribir		

Ejemplo de Matriz de Acceso

2.5. Implementación de la Matriz de Acceso

2.5.1. Tabla Global

- ★ Se trata de la implementación más simple
- ★ Está compuesta por un conjunto de tripletas ordenadas:
  - **<dominio, objeto, conjuntoDerechos>.**
- ★ Cada vez que se ejecuta una operación M sobre un objeto Oj dentro del dominio Di, se analiza la tabla global en busca de una tripleta **<Di, Oj, Rk> con M ∈ Rk.**
- ★ Si se encuentra dicha tripleta, se permite que la operación continué; en caso contrario; se genera una condición de excepción(error).



Ejemplo de Tabla Global

2.5.2. Listas de Acceso para los Objetos

El segundo método para implementar a Matriz de Acceso se trata de una lista de acceso de un objeto.

- Las entradas vacías pueden descartarse.
- La lista resultante para cada objeto estará compuesta por una serie de parejas ordenadas: **<dominio, conjuntoDerechos>,** los cuales definen todos los dominios que tengan un conjunto de derechos de acceso no vacío para dicho objeto.

2.5.3. Listas de Capacidades para los Dominios

El tercer método se trata de una lista de capacidades para los dominios, esto es una lista de objetos junto con las operaciones permitidas sobre esos objetos.

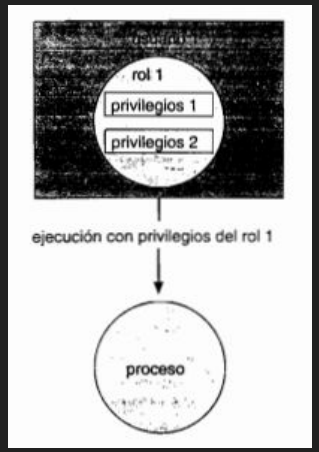
- Cada objeto suele representar mediante su dirección o nombre físico, denominada **CAPACIDAD.**
- Esta lista está asociada con un dominio, pero un proceso que se ejecute en el dominio no puede acceder nunca directamente a ella.

2.6 Control de Acceso

A cada archivo y directorio se le asigna un propietario, un grupo o una lista de usuarios y para cada una de estas entidades se asigna una información de **CONTROL DE ACCESO.**

Un ejemplo de esto es el de ***Solaris 10:***

> Solaris 10 amplía el sistema de protección disponible en el S.O “Sun Microsystems”, añadiendo de manera explícita, el principio del mínimo privilegio (revisado en diapositivas previas), mediante el control de acceso basado en roles(RBAC).



Control de Acceso basado en Roles en Solaris 10

Solaris 10

2.7. Revocación de derechos de Acceso

En un sistema de protección dinámico, puede que necesitamos en ocasiones revocar derechos de acceso a objetos compartidos por diferentes usuarios.

Con esto, pueden surgir diversos tipos de revocaciones:

Inmediata o Diferida.- Surge la pregunta ¿La revocación tiene lugar inmediatamente o se produce de forma diferida?

Selectiva o General.- Cuando se revoca un derecho de acceso a un objeto, ¿Afecta a todo los usuarios que tienen derecho de acceso a ese objeto?

Parcial o Total.- ¿Se puede revocar un subconjunto de los derechos asociados con un objeto o se debe revocar todos los derechos de acceso al objeto?

Temporal o Permanente.- ¿Se puede revocar el acceso permanentemente, o se puede revocar el acceso y luego obtenerlo de nuevo?

2.8. Sistemas Basados en Capacidad

2.8.1. Hydra

- Es un sistema de protección basado en capacidades que proporciona una flexibilidad.
- El sistema conoce e interpreta un conjunto fijo de posibles derechos de acceso.
- Esos derechos incluyen formas básicas de acceso tales como el derecho de leer, escribir o ejecutar un segmento de memoria.
- Un usuario puede declarar otros derechos.
- La interpretación de los derechos definidos por el usuario se lleva a cabo exclusivamente por el programa de usuario.
- Estas características constituyen un avance significativo en la tecnología de protección.

2.8.2. Sistema CAP de Cambridge

- Ha adoptado un enfoque distinto para la implementación del mecanismo de protección basado en capacidades.
- Es más simple y superficialmente menos potente que el de Hydra.
- Dispone de dos tipos de capacidades:
    - **Capacidad de Datos.-** Puede utilizarse para proporcionar acceso a los objetos, pero los únicos derechos proporcionados son los derechos normales de lectura, escritura y ejecución de los segmentos de almacenamiento individuales asociados con el objeto.
    - **Capacidad de Software.-**Está protegido, pero no interpretado, por el microcódigo del sistema. Para interpretar estas capacidades se utiliza un procedimiento ***"protegido"*** o ***"privilegiado",*** el cual puede ser escrito por un programador de aplicaciones como parte de un subsistema.

2.9. Protección Basada en el Lenguaje

El grado de protección que se proporciona en los sistemas informáticos existentes suele conseguirse mediante un kernel del S.O. que actúa como agente de seguridad para inspeccionar y validar cada intento de acceder a un recurso protegido.

A medida que ha aumentado la complejidad de los S.O y particularmente a medida que estos han tratado de proporcionar interfaces de usuario de mayor nivel, los objetivos de la protección se han hecho mucho más “refinados”.

3. Virtualización

3.1. Requerimientos para la Virtualización

Existen dos métodos para la virtualización:

Hipervisor de tipo 1 (Máquina Virtual).- El cual es el S.O. como tal, ya que es el único programa que se ejecuta en modo Kernel. Su trabajo es soportar varias copias del hardware actual(Máquinas Virtuales), de una manera similar a los procesos que soporta un S.O. normal.

Hipervisor de tipo 2.- Se trata de solo un programa de usuario que se ejecuta ya sea en Windows o Linux e "interpreta" el conjunto de instrucciones de la máquina, el cual también crea una máquina Virtual

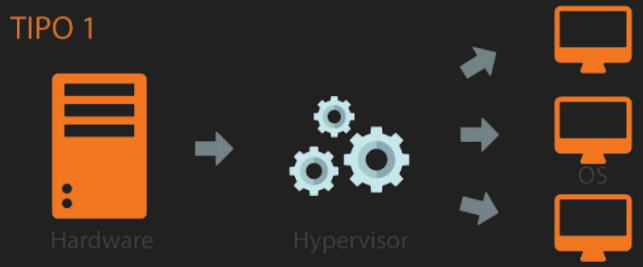
3.2. Hipervisores de tipo 1

La capacidad de virtualización es una cuestión importante.

En la figura inferior, se encuentra un hipervisor de tipo 1 que soporta una máquina virtual.

Al igual que todos los hipervisores de tipo 1, se ejecuta en modo de kernel.

La máquina virtual se ejecuta como un proceso de usuario en modo de usuario y, como tal, no puede ejecutar instrucciones sensibles. Además, ejecuta un sistema operativo invitado que piensa que se encuentra en modo de kernel, aunque desde luego se encuentra en modo de usuario. A éste le conoce como **kernel virtual**.

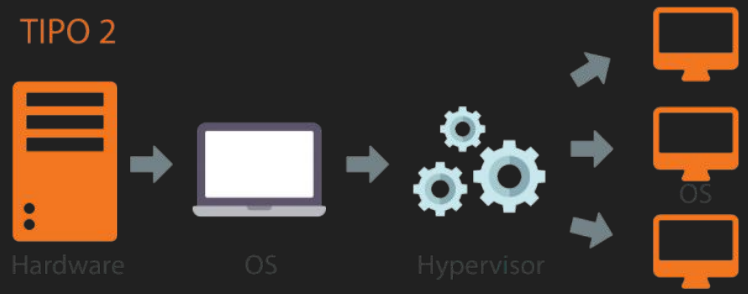


- La máquina virtual también ejecuta procesos de usuario, los cuales creen que se encuentran en modo de usuario.

3.3. Hipervisores de tipo 2

Para poder crear un sistema de máquina virtual es es necesario que haya Virtualización disponible. Anteriormente, no se podría ejecutar un sistema operativo completo en una máquina virtual debido a que sólo se ignorarán las instrucciones sensibles, y el sistema fallaría.

Sin embargo,, lo que ocurrió fue la invención de lo que ahora se conoce como hipervisores de tipo 2, como se muestra en la figura inferior.



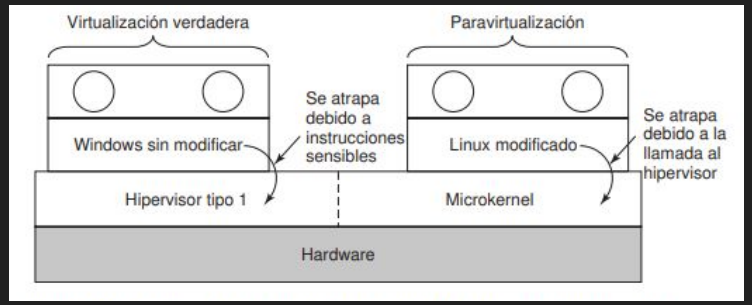
- Se puede esperar que las CPU s con Virtualización posean un rendimiento mucho mayor que las técnicas de software que se utilizan en los hipervisores de tipo 2.

3.4. Paravirtualización

- Los hipervisores de tipo 1 y tipo 2 funcionan con S.O. invitados que no estén modificados, no obstante, tienen que hacer un gran esfuerzo por obtener un rendimiento razonable.
- Un método distinto que se está haciendo popular es modificar el código fuente del S.O., de manera que en lugar de ejecutar instrucciones sensibles, realiza llamadas al hipervisor.

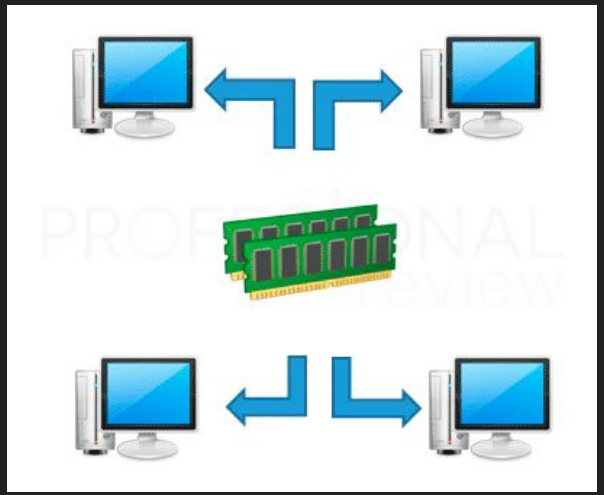
Sin duda, el S.O. invitado actúa como un programa de usuario que realiza llamadas al sistema operativo (el hipervisor). Cuando se utiliza este método, el hipervisor debe definir una interfaz que consiste en un conjunto de llamadas a procedimientos que los S.O. invitados puedan utilizar.

Estas llamadas forman una **API (Interfaz de Programación de Aplicaciones)**, aun cuando es una interfaz para que la utilicen los sistemas operativos invitados, y no los programas de aplicación.



3.5. Virtualización de la Memoria

Casi todos los S.O. modernos soportan la memoria virtual, la cual es una asignación de las páginas en el espacio de direcciones virtuales a las páginas de la memoria física. Esta asignación se define mediante tablas de páginas. Por lo general la asignación se establece en movimiento, al hacer que el sistema operativo establezca un registro de control en la CPU que apunte a la tabla de páginas de nivel superior. L a virtualización complica de manera considerable la administración de la memoria.

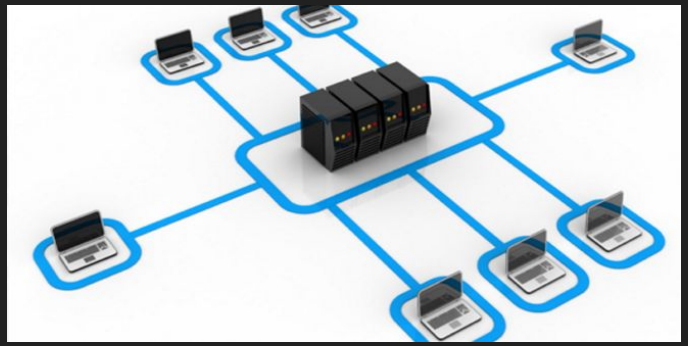


3.6. Virtualización de la E/S

- Por lo general, el S.O. invitado empieza con un sondeo del hardware para averiguar qué tipos de dispositivos de E/S están conectados.
- Dichas sondas producirán interrupciones en el hipervisor.

Ahora bien, **¿Qué debe hacer el hipervisor?**

Un método considerable, es que reporte de vuelta que los discos, las impresoras y demás dispositivos son los que realmente tiene el hardware. Después el invitado cargará drivers para estos dispositivos y tratará de utilizarlos. Cuando los drivers de los dispositivos traten de realizar operaciones de E/S, leerán y escribirán en los registros de dispositivo del hardware del dispositivo correspondiente. Estas instrucciones son sensibles y se atraparán en el hipervisor, que podría entonces copiar los valores necesarios que entran y salen de los registros de hardware, según sea necesario.



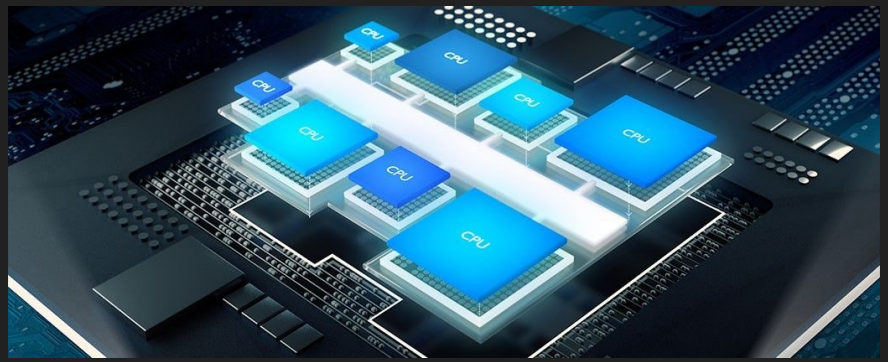
3.7. Dispositivos Virtuales

- El problema principal de la virtualización es que muchas aplicaciones dependen de otras tantas aplicaciones y bibliotecas, que a su vez dependen de muchos otros paquetes de software, y así en lo sucesivo.
- Además, puede haber dependencias en versiones específicas de los compiladores, lenguajes de secuencias de comandos y en el sistema operativo.
- Ahora que están disponibles las máquinas virtuales, un desarrollador de software puede construir con cuidado una máquina virtual, cargarla con el sistema operativo, compiladores, bibliotecas, etc y congelar la unidad completa, lista para ejecutarse.
- Esta imagen de la máquina virtual se puede colocar en un CD-ROM o en un sitio Web para que los clientes la instalen o descarguen.
- Este método implica que sólo el desarrollador de software tiene que conocer todas las dependencias.
- A estas máquinas virtuales “empaquetadas y listas para usarse” se les conoce comúnmente como dispositivos virtuales.

3.8. Máquinas Virtuales en CPUs de Multinúcleo

La combinación de las máquinas virtuales y las CPUs de multinúcleo abre todo un nuevo mundo, donde el número de CPUs disponibles se puede establecer en el software.
Ejemplo:

Si tenemos cuatro núcleos y cada uno de ellos se puede utilizar para ejecutar, ocho máquinas virtuales, se puede configurar una sola CPU como una multicomputadora de 32 nodos si es necesario, sin embargo también puede tener menos CPUs dependiendo de las necesidades del software.
Nota: Nunca antes había sido posible que un diseñador de aplicaciones seleccionara primero cuántas CPU s quiere, para después escribir el software de manera acorde. Esto representa una nueva fase en la computación.



3.9. Licencias

- Las licencias de la mayoría del software son por cada CPU(individual). En otras palabras, cuando un usuario adquiere un programa, tiene el derecho de ejecutarlo sólo en una CPU .
- La pregunta que afecta en este caso es: **¿Acaso este contrato le otorga el derecho de ejecutar el software en varias máquinas virtuales, todas las cuales se ejecutan en la misma máquina?**
- Este problema es mucho peor en las empresas que tienen una licencia que les permite tener **n** máquinas ejecutando el software al mismo tiempo, en especial cuando aumenta y disminuye la demanda de las máquinas virtuales.
- En algunos casos, los distribuidores de software han colocado una cláusula explícita en la licencia, en la que se prohíbe al concesionario ejecutar el software en una máquina virtual o en una máquina virtual no autorizada.

Conclusiones

- Sin duda todo este tema de la seguridad, protección y virtualización es un aspecto fundamental dentro de los S.O. ya que puede prevenir al usuario, a los desarrolladores, inclusive a grandes empresas de alguna pérdida de información, robo de datos, abuso de privacidad, etc.
- Es muy importante el comprender el origen de cada intruso, así como las técnicas para contenerlos de manera eficiente y que no se pueda reproducir dentro de nuestras máquinas como es el caso de los gusanos informáticos.
- Las técnicas tanto para esta prevención como para virtualizar un dispositivo son fundamentales de comprender, tanto para un programador, como para un usuario normal, ya que es evidente que la tecnología está avanzando a un paso agigantado y conforme esta tecnología crece, también pueden crecer sus amenazas y sus opciones para enfrentarlas.