**Parte 1: Conceptos teóricos**

1. **Defina virtualización. Investigue cuál fue la primer implementación que se realizó.**

**Virtualización**:

Es el efecto de abstraer los recursos de un computador, proporcionar acceso lógico a recursos físicos.

creaciones través de software de una versión virtual de un recurso tecnológico. Abstracción de los recursos de una computadora, hypervisor que crea una capa de abstracción entre el hardware y el sistema operativo invitado (de la vm).

En simples palabras, es el proceso mediante el que se crea una versión virtual, en lugar de una física. La virtualización se puede aplicar a computadoras, sistemas operativos, dispositivos de almacenamiento de información, aplicaciones o redes.

Primera implementación, IBM CP-40.

1. **¿Qué diferencia existe entre virtualización y emulación?**

Con la virtualización el sistema huésped se ejecuta realmente en el HW subyacente, el hypervisor solo media en los accesos de la vm al hardware real, solo se puede montar un SO compatible con la misma arquitectura del procesador.

El emulador es un programa que simula la funcionalidad de otro programa o un componente de hardware. Dado que implementa la funcionalidad por software proporciona una gran flexibilidad.El programa emulado puede incluso ser escrito para una arquitectura diferente a la que se ejecuta.

1. **Investigue el concepto de hypervisor y responda:**
2. **¿Qué es un hypervisor? Investigue cuándo se realizó la primer implementación de está tecnología.**

Es una plataforma que permite aplicar diversas técnicas de control de [virtualización](https://es.wikipedia.org/wiki/Virtualizaci%C3%B3n) para utilizar, al mismo tiempo, diferentes [sistemas operativos](https://es.wikipedia.org/wiki/Sistema_operativo) (sin modificar o modificados, en el caso de [paravirtualización](https://es.wikipedia.org/wiki/Paravirtualizaci%C3%B3n)) en una misma computadora.

Un Hypervisor es un monitor de máquina virtual, VMM. UN hipervisor corre una o más Virtual Machines en una host machine. Fueron originalmente desarrollados en principio de los años 70. IBM

1. **¿Qué beneficios traen los hypervisors? ¿Cómo se clasifican?**

Los beneficios son: permiten reducir costos, al correr multiples SO al mismo tiempo, si un SO colapsa los otros siguen trabajando.

**Los Hypervisors se clasifican en 2 tipos**

* Tipo1 (nativo, bare metal). Corren directamente en el HW del host para controlarlo y manejar a los SO Guest.Está escrito en el firmware
* Tipo 2 (hosted): en un hipervisor que se ejecuta sobre el SO del host para ofrecer su funcionalidad

El hypervisor tipo 1 no puede correr en una arquitectura sin virtualización el otro si.

1. **Indique por qué un hypervisor de tipo 1 no podría correr en una arquitectura sin tecnología de virtualización. ¿Y un hypervisor de tipo 2 en hardware sin tecnología de virtualización?**

Porque corre directamente sobre el hardware. El hardware debe proveer algún mecanismo para distribuir sus recursos y poder ejecutar dos s.o al mismo tiempo.   
La virtualización por software es decir administrada por un s.o no existía.

Un hipervisor de tipo 1 no podría correr en una arquitectura sin tecnologia de virtualizacion ya que cada máquina virtual se ejecuta como un proceso de usuario, sobre el hardware real que se particiona para poder ser compartido por todas las MV.

En el caso del hipervisor de tipo 2, el hardware es emulado por lo que cada maquina “corre” sobre un hardware específico más allá del que realmente posee la pc.

1. **Investigue el concepto de paravirtualización y responda:**
2. **¿Qué es la paravirtualización?**

La [paravirtualización](http://www.jonathanecheverria.com/tag/paravirtualizacion) es un sistema virtualizador mediante el cual, las instrucciones de la MV(Máquina Virtual) se ejecutan directamente en el procesador físico, puesto que emplea sistemas operativos modificados para ello.

La intención de la interfaz modificada es reducir la porción del tiempo de ejecución del usuario, empleado en operaciones que son sustancialmente más difíciles de ejecutar en un entorno virtual, comparado con un entorno no virtualizado. La paravirtualización provee filtros especialmente definidos para permitir a los invitados y al anfitrión hacer peticiones y conocer estas tareas, que de otro modo serían ejecutadas en el dominio virtual (donde el rendimiento de la ejecución es peor.) Por lo tanto, una plataforma de paravirtualización exitosa puede permitir que el monitor de la máquina virtual (VMM) sea más simple (por traslado de la ejecución de tareas críticas desde el dominio virtual al anfitrión de dominio), y/o que reduzca la degradación del rendimiento global de la ejecución de la máquina dentro del anfitrión virtual.

La paravirtualización requiere que el sistema operativo invitado sea portado de manera explícita para la API. Una distribución de un sistema operativo convencional que no soporte paravirtualización no puede ser ejecutada ni visualizada en un monitor de máquina virtual VMM

1. **¿Sería posible utilizar paravirtualización en sistemas operativos como Windows o iOS? ¿Por qué?**

Ni windows ni IOS se pueden correr paravirtualizados porque no se puede modificar el código fuente como lo requiere un SO para ser paravirtualizado, SI EXISTEN unos drivers de WIndows que lo modifican para poder correrlos, pero no vienen nativos.

Sería posible si los propietarios de dichos SO proveen de las versiones paravirtualizadas de dichos programas, ya que no son open source, por lo cual no podemos ver su código para modificarlo; o bien instalando drivers paravirtualizados, lo que nos permitirá tener un SO híbrido.

1. **Mencione algún sistema que implemente paravirtualización.**

* FreeBSDuntil version 11.0
* GNU/Hurd (gnumach-1-branch-Xen-branch)
* Linux, paravirtualization integrated in 2.6.23, patches for other versions exist
* MINIX
* NetBSD
* OpenSolaris (See The Xen Community On OpenSolaris)
* OZONE (has support for Xen v1.2)
* Plan 9 from Bell Labs
* OpenBSD, as of 5.9[58]

1. **Defina VMI.**

VMI resuelve una serie de problemas de seguridad desde fuera del sistema operativo invitado sin depender de la funcionalidad que el malware avanzado puede volver poco fiable. El enfoque funciona auditando el acceso de áreas de memoria sensibles usando soporte HW en invitados de una manera discreta (o quizás mejor: con una sobrecarga mínima) y permite que el software de control se ejecute dentro de una VM dedicada para permitir o denegar intentos de acceso a memoria sensible heurística de seguridad

1. **¿Qué beneficios trae con respecto al resto de los modos de virtualización?**
2. **Investigue si VMI podría correr sobre hypervisors de tipo 1 ó 2, y justifique por qué.**

Sobre tipo 2 no tendría sentido, si en cambio sobre tipo 1 ya que permite al so guest ejecutar sus intrucciones sensibles a través de llamadas al hipervisor que provee VMI. En el tipo 2 el so guest ejecuta las instrucciones sensibles de manera tradicional ya que no estan modificados para hacer llamadas a funciones del hipervisor por lo que no tiene sentido vmi, todas las instrucciones sensibles las captura el hipervisor.

1. **Investigue sobre containers en el ámbito de la virtualización y responda:**
2. **¿Qué son?**

Es una tecnología liviana de virtualización a nivel de sistema operativo que permite ejecutar múltiples sistemas aislados. Sus instancias se ejecutan en espacio de usuario. Todas comparten el mismo kernel. No necesita software de virtualización o hypervisor.USA CGROUPS, NAMESPACE ISOLATIONS Y CHROOT. Es virtualización a nivel OS. Proyecto comenzado a mediados del 2008.

Cada container tiene:

* su propio nombre y dominio
* sus propias interfaces de red (junto con sus Ips)
* Tiene sus propios filesystems
* Tiene su propio espacio de nombres de procesos ID (PIDs) e IPC
* Provee aislación (seguridad y uso de recursos)

1. **¿Dependen del hardware subyacente?**

No.

1. **¿Qué lo diferencia por sobre el resto de las tecnologías estudiadas?**

Los contenedores y las máquinas virtuales tienen beneficios similares de aislamiento de recursos y asignación, pero funcionan de manera diferente porque los contenedores virtualizan el sistema operativo en el lugar del hardware. Los contenedores son más portátiles y eficientes.

1. **Investigue qué funcionalidades del kernel Linux permiten la implementación de containers.**

LXC funciona aprovechando una funcionalidad del núcleo Linux llamada cgroups, que está presente en el kernel desde la versión 2.6.29 (y por lo tanto en todas las versiones modernas 3.x). LXC es por lo tanto virtualización a nivel del sistema operativo.

lxc-create permite crear un contenedor.

lxc-create -n “nombre contenedor” -t “template”

Por ejemplo: **lxc-create -n lxcdebian -t debian**

**Comandos relacionados a containers**

lxc-start -n lxcdebian -d para iniciar un container

lxc-info -n “nombre del contenedor” para obtener info sobre un container running/stopped

lxc-ls –active para ver containers ejecutando

lxc-ls -fancy para comprobar el estado de un container

lxc-clone clona un container

lxc-console -n “nombre del contenedor” para entrar en la consola de un container

lxc-stop -n “nombre del container”para detener el contenedor

Para crear un container a partir de un archivo de configuración:

* lxc-create -t debian -f file.conf -n “nombre del contenedor”

Para cambiar root password

* chroot en el filesystem del contenedor
  + chroot /var/lib/lxc/contenedor/rootfs
* Comando passwd e ingresar la nueva password del root
* También es posible hacerlo modificando el archivo “shadow” del containers

Para montar otro file system agregar en el archivo de configuración del container (var/lib/lxc/contenedor/config)

**Parte 2: Control Groups, Namespaces y Containers**

**CHROOT**: Es una forma de aislar aplicaciones del resto del sistema.Introducido en la versión 7 de UNIX en 1979.Cambia el directorio raíz aparente de un proceso. Al entorno virtual creado por chroot a partir de la nueva raíz del sistema se la conoce como “JAIL CHROOT”. NO PODEMOS acceder a archivos y comandos fuera de ese directorio. En algunos casos suele ser conveniente restringir la cantidad de información a la que un proceso puede acceder. Uno de los métodos más simples para aislar servicios es chroot, que consiste simplemente en cambiar lo que un proceso junto con sus hijos consideran que es el directorio raíz, limitando de esta forma lo que pueden ver en el sistema de archivos. En esta sección de la práctica se preparará un árbol de directorios que sirva como directorio raíz para la ejecución de una shell.

“chroot /directorio comando” ← Para que chroot funcione correctamente en Linux se deben montar ciertos sistemas de archivos API que necesita el sistema operativo.

* Mount –bind /dev/ target/dev/
* mount -bind /proc/ target/proc/
* mount –bind /sys/ target/sys/

Por último chroot target → target es el directorio que elegimos.

**1. Crear un subdirectorio llamado sobash dentro del directorio root. Intente ejecutar el comando chroot /root/sobash. ¿Cuál es el resultado? ¿Por qué se obtiene ese resultado?**

chroot: failed to run command ‘/bin/bash’: No such file or directory

Esto se debe a que al cambiar el directorio del proceso, éste no conoce el comando que se utiliza para levantar una consola ya que se ubica en la carpeta /bin.

**2. Copiar en el directorio anterior todas las librerías que necesita el comando bash. Para obtener esta información ejecutar el comando ldd /bin/bash. ¿Es necesario copiar la librería linux-vdso.so.1? ¿Por qué? Dentro del directorio anterior crear las carpetas donde va el comando bash y las librerías necesarias. Probar nuevamente. ¿Qué sucede ahora?**

No es necesario. Esto se debe a que la librería forma parte del kernel, no es un módulo.

Lo que sucede es que se crea un nuevo proceso cuyo directorio ‘/’ es ‘/root/sobash’, es decir que se crea una nueva terminal. Esta terminal posee unos pocos comandos, ya que el resto no fueron copiados a la carpeta ‘/root/sobash/bin’.

**3. ¿Puede ejecutar los comandos cd "directorio" o echo? ¿Y el comando ls? ¿A qué se debe esto?**

Es posible ejecutar los comandos cd y echo, ya que estan incluidos en el código de bash. No es posible ejecutar el comando ls ya que el directorio /root/sobash/bin no contiene el archivo /bin/ls (es un binario externo)

1. **¿Cuál es la finalidad de la herramienta debootstrap?**

Debian posee un método simple que permite instalar un sistema base DEBIAN en el subdirectorio de otro sistema instalado. Puede ser instalada y ejecutado desde otro SO y otra arquitectura (*cross-debootstrapping*). Lo básico para hacerla

La sintaxis de debootstrap para crear un sistema base es: debootstrap **<suite> <target> <mirror>.** Como suite se utiliza la versión stable de debian, stable, como target el directorio donde se aloja el árbol de directorios y como mirror: <http://httpredir.debian.org/debian>.

**11. ¿Puede ver los procesos que corren en el sistema operativo base? ¿Por qué?**

No es posible, ya que esta terminal esta corriendo en “otro sistema operativo” con sus propios recursos y procesos.

**12. Los contenidos de los directorios /home, ¿son iguales? ¿Por qué?**

No lo son, por el mismo motivo anteriormente mencionado

**13. ¿Qué desventajas encuentra en el uso de chroot como medida de seguridad?**

No tendría sentido utilizarla como medida de seguridad, es una herramienta demasiado grande para una funcionalidad tan simple, o algo asi dijo Agustin Parmisano.

**14. Si se ejecuta un servidor HTTP en el SO base, ¿es posible ejecutar otro servidor HTTP escuchando en el mismo puerto en el entorno chroot? ¿Por qué?**

Esto no es posible.

* **CONTROL GROUPS**

Control Groups, cgroups, es un mecanismo que permite organizar procesos en forma jerárquica y distribuir los recursos del sistema (CPU, Memoria, I/O) a lo largo de la jerarquía en una manera controlada y configurable. Disponible desde la versión 2.6.24 del kernel. La versión 2 de CG se lanzó con el kernel 4.5 en 2016. Permite un control de “grano fino” en la alocación, priorización, denegación y monitoreo de los recursos del sistema.

* Organizado jerárquicamente, como los procesos, y los hijos heredan atributos
* Pueden existir muchas jerarquías diferentes de cgroup
* Cada jerarquía se asocia a uno o más subsistemas
* Un subsistema representa un único recurso: tiempo de CPU, memoria, etc
* Compuesto por 2 partes:
  + **Core**: responsable de organizar jerarquicamente.
  + **Controller**: responsable de distribuir los recursos del sistema a lo largo de la jerarquías
* Cada proceso del sistema solo puede pertenecer a un cgroup dentro de una jerarquias

Se implementa como un pseudo filesystem, los procesos y todos sus threads, pertenecen a un solo “cgroup”. Los procesos no son conscientes de los límites aplicados por un cgroup. Una vez definidos los grupos, se le agregan los IDs de los procesos.

Libcgroup: tools para administrar los cgroups

Ejemplo de cgroup, creando dos grupos dentro del subsistema cpu, llamadas cpu alta y cpu baja

“mkdir /sys/fs/cgroup/cpu/”nombrecgroup”

Para indicar a cada uno de los cgroups creados en el paso anterior el porcentaje de cpu máximo que cada uno puede utilizar, se pone el valor en el .shares

“echo 7 /sys/fs/cgroup/cpu/cpualta/cpu.shares”

Para ligar el proceso a un cgroup

“echo process\_pid >/sys/fs/cgroup/cpu/nombrecgroup/task”

**4. Crear dos grupos dentro del subsistema cpu llamadas cpualta y cpubaja. Controlar que se hayan creado tales directorios y ver si tiene algún contenido**

mkdir /sys/fs/cgroup/cpu/cpualta

mkdir /sys/fs/cgroup/cpu/cpubaja

**11. Desde otra terminal observar como se comporta el uso de la CPU. ¿Qué porcentaje de CPU recibe cada uno de los procesos?**

En la xterm que fue agregado el proceso cpualta (azul) el porcentaje de cpu es de alrededor del 70%, y en la otra al 30% (varía alrededor de ese número).

**15. Ejecutar nuevamente el comando md5sum /dev/urandom & en cada una de las terminales.** ¿Qué sucede con el uso de la CPU? ¿Por qué?

Ahora el uso de la cpu es del 50% en ambos casos. Por balanceo de carga

**16. Si en termbaja ejecuta el comando taskset -c 0md5sum /dev/urandom & (deben quedar 3  
comandos md5 ejecutando a la vez, 2 en el termbaja). ¿Qué sucede con el uso de la CPU?  
¿Por qué?**

El uso de la cpu es del 25% aproximadamente. Por balanceo de carga

* **NAMESPACE**: namespace isolation permite abstraer un recurso global del sistema para que los procesos dentro de ese namespace piensen que tienen su propia instancia aislada de ese recurso global. Las modificaciones a un recurso quedan contenidas dentro del “namespace”

Linux provee los siguientes namespaces:

* **IPC**: System V IPC,cola de mensajes POSIX.
* **Network**:dispositivos de red,pilas, puertos, etc.
* **Mount**: puntos de montaje (permite aislar la tabla de montajes,usado típicamente con chroot)
* **PID**: IDs de procesos (Da la posibilidad de tener múltiples árboles de procesos anidados y aislados)
* **Users**: IDs de usuarios y grupos

**1. Explique el concepto de Namespaces. ¿Cuáles son los posible NameSpaces disponibles?**

Namespace Isolation: Permite abstraer un recurso global del sistema para que los procesos dentro de ese “namespace” piensen que tienen su propia instancia aislada de ese recurso global

Entre los namespaces provistos por Linux:

IPC: System V IPC, cola de mensaje POSIX

Network: dispositivos de red, pilas, puertos, etc

Mount: puntos de montaje

PID: IDs de procesos

User: IDs de usuarios y grupos

UTS: HostName y nombre de dominio

**2. Crear dos namespaces que se llamarán nsserver y nsclient**

# ip netns add nsserver

# ip netns add nsclient

**3. Crear dos interfaces virtuales, tipo veth (se generan de a pares y están conectadas por medio de una tubería), que se llamarán vethsrv y vethcli:**

# ip link add vethsrv type veth peer name vethcli

**4. A continuación se deben agregar las veth a los namespaces (vethsrv a nsserver y vethcli a nsclient)**

# ip link set vethcli netns nsclient

# ip link set vethsrv netns nsserver

**5. Asignarle IPs a las interfaces virtuales de cada uno de los namespaces (10.10.10.1/24 a la interface en nsserver y 10.10.10.2/24 a la existente en nsclient)**

# ip netns exec nsserver ip addr 10.10.10.2/24 dev vethsrv

# ip netns exec nsclient ip addr 10.10.10.1/24 dev vethcli

**6. Si el comando anterior indica que las interfaces están bajas (DOWN) se deben activar:**

# ip netns exec nsserver ip link set vethsrv up

# ip netns exec nsclient ip link set vethcli up

**9. Sin cerrar los comandos nc, ejecutar en el entorno global el comando ss -nat. ¿Puede ver el estado en que se encuentra el puerto 9043? ¿Por qué? Ejecute el mismo comando pero con la opción -N "namespace". ¿Es posible ver alguna información con respecto a ese puerto?**

No se puede ver el estado en que se encuentra el puerto, ya que en el contexto que se ejecuta el ss es el global y este puerto esta siendo usado pero en el contexto de otro namespace.

Es posible ver la información del puerto ejecutando el comando como sudo.

**10. Ejecutar un bash dentro de uno de los nameservers y ver qué interfaces de red están activas.**

ip netns exec nsclient bash

State Recv-Q Send-Q Local Address:Port Peer Address:Port

ESTAB 0 0 10.10.10.2:9043 10.10.10.1:50916

ip netns exec nsserver bash

State Recv-Q Send-Q Local Address:Port Peer Address:Port

ESTAB 0 0 10.10.10.1:50916 10.10.10.2:9043

**Containers**

**6. Utilizar el comando lxc-ls -f para comprobar el estado de los containers. ¿En qué estado**

**se encuentra el container sodebian?**

sudo lxc-ls -f

NAME STATE AUTOSTART GROUPS IPV4 IPV6

sodebian RUNNING 0 - 10.0.3.168 -

**8. ¿Tiene la password para ingresar container? En caso de no tenerla modificar la password**

**de root usando chroot**

**Nota: con el comando lxc-attach es posible acceder a un container sin tener la password**

sudo lxc-attach --name=sodebian

**9. Ejecutar el script utilizado en los ejercicios anteriores y responder:**

**(a) ¿Puede acceder al /home del sistema operativo base?**

**(b) ¿Es posible ver los procesos que están corriendo en el sistema operativo base?**

No es posible acceder al /home del sistema operativo base, los archivos no son los mismos.

No es posible ver los procesos que están corriendo en el sistema operativo base.

**11. ¿Qué interfaces de red existen en el container? Analizar el contenido del archivo config en el directorio /var/lib/lxc/sodebian/.**

lxc.network.type = veth

lxc.network.link = lxcbr0

lxc.network.flags = up

lxc.network.hwaddr = 00:16:3e:92:15:2e

lxc.rootfs = /var/lib/lxc/sodebian/rootfs

lxc.rootfs.backend = dir

# Common configuration

lxc.include = /usr/share/lxc/config/debian.common.conf

# Container specific configuration

lxc.tty = 4

lxc.utsname = sodebian

lxc.arch = amd64