Universidad de Buenos Aires

Facultades de Ciencias Económicas,

Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final

Seguridad en virtualización

Autor: Juan Pablo Benedetti

Tutora: Sabrina Irisarri González Deibe

Año de Presentación: 2019

Cohorte: 2016

Declaración Jurada

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

FIRMADO

Juan Pablo Benedetti

DNI: 25360888

Resumen

El presente trabajo investiga la información relacionada con la virtualización, en particular con las máquinas virtuales. El foco está puesto en los temas relacionados con seguridad informática. Aquí se hace un análisis de su historia, sus tipos, características, ventajas y desventajas. Se presentan sus vulnerabilidades y distintos ataques que han surgido. Se describen las recomendaciones a tener en cuenta de dos reconocidas instituciones (NIST e ISO/IEC).

Dada la relevancia actual de la virtualización, consideramos que es fundamental comprender todas las características de la virtualización y sus problemáticas. Este trabajo intenta ser un aporte para entender esta herramienta fundamental en el mundo de la informática.

Índice

Índice de figuras

1 Introducción

La virtualización es una herramienta muy valiosa en el mundo de las Tecnologías Informáticas. Abastece recursos de cómputo, de red y de almacenamiento de manera flexible y según la demanda. Permite utilizar la infraestructura de manera más efectiva, logrando optimizar costos [1]. Facilita la abstracción del *hardware* y divide los recursos para compartirlos en contenedores lógicos o máquinas virtuales (MVs). Otorga flexibilidad para adaptarse a los cambios que necesitan las empresas cuyos requerimientos evolucionan constantemente debido, principalmente, a demandas y desafíos del negocio y nuevas regulaciones.

En los últimos años, la virtualización se ha generalizado en todo tipo de organizaciones [1]. Según algunas estimaciones, en 2016 ya era usada por el 76% de las organizaciones [2]. Si bien se comenzó a usar en ambientes no productivos (por ejemplo, desarrollo, pruebas y capacitación), pronto su uso se extendió a los ambientes de producción. El motivo fue que ofrecía el desempeño y la estabilidad requerida, reducía gastos en capital (*capex*[1], al reducir la cantidad de servidores y el consiguiente espacio en el centro de datos [3]), reducía costos de operación (*opex*[2], gracias a la mejora en la productividad, simplificación en las tareas de gestión, menor consumo eléctrico, etc. [3]) y era más ágil [4]. Para la consultora *Gartner*, según muestra en sus ciclos de sobreexpectación[3] (*hype cycle*), las MVs han alcanzado la meseta de la productividad (*Plateau of Productivity*) por lo menos desde 2012 [5].

La amplia adopción de la computación en la nube (*cloud computing*) [6] no le resta importancia a la virtualización, sino, por el contrario, aumenta su

[1] Por la contracción del inglés *capital expenditure*.

[2] Por la contracción del inglés *operational expenditure*.

[3] Un ciclo de sobreexpectación es una herramienta gráfica que representa la madurez, la adopción y el uso de una determinada tecnología, mediante un eje que indica el nivel de visibilidad o expectativa y otro eje con el tiempo. El término fue acuñado por la consultora Gartner que lo usa desde 1995 para mostrar cómo nuevas tecnologías suelen despertar un entusiasmo y expectativas exageradas, luego de los cuales aparece una desilusión que genera una menor visibilidad hasta un punto a partir del cual la tecnología se va consolidando y alcanza lo que denomina una meseta de productividad.

interés y relevancia. Esto se debe a que la virtualización es una tecnología central y clave para la infraestructura de la nube [7]. Para Khalil et al. [6] es su columna vertebral. Según Yin et al. [8], con la adopción de la computación en la nube, la virtualización se vuelve ubicua. El uso de esta tecnología en la nube agrava varios de los problemas de la virtualización, ya que organizaciones, empresas y gobiernos pueden potencialmente compartir un mismo recurso físico (donde corre la virtualización) con atacantes [6]. Este escenario, donde varios usuarios o grupos de usuarios ejecutan sus procesos en un mismo dispositivo físico, se lo denomina *multi-tenancy* (multiusuario). Aquí cada usuario o grupo de usuarios pueden ser de distintas organizaciones. En *multi-tenancy* se usa la virtualización para que distintos clientes de la nube compartan un mismo *hardware* [9]. Esta compartición es la fuente de varios problemas de seguridad (algunos ejemplos de esta problemática se presentan en los trabajos de Khreishah et al. [6], Zhang et al. [9], Ristenpart et al. [10], Jasti et al. [11], D'Antonio et al. [12] y Varadarajan et al. [13]).

Teniendo en cuenta lo expresado más arriba, es muy importante conocer esta tecnología y saber qué aspectos se deben tener en cuenta desde el punto de vista de la seguridad antes de adoptar su uso. En este trabajo se hará una introducción a la virtualización y a los distintos tipos actualmente en uso. Se hará foco en un tipo de virtualización, la de servidores. Se hablará de las ventajas y desventajas de su uso, se analizarán los problemas y cuestiones a tener en cuenta desde el punto de vista de la seguridad informática. Se verán ataques y vulnerabilidades que surgieron a medida que se iba extendiendo su uso y aumentaba el interés en el estudio de esta tecnología.

1.1 Organización de este trabajo

En el capítulo 2 se introduce el tema de la virtualización, su historia, se presenta al hipervisor, sus tipos y características. En el capítulo 3 se explican las ventajas y desventajas de su uso, tanto las generales como las específicas del área de seguridad. En el capítulo 4 se presentan y describen los distintos tipos de virtualización existentes. En el capítulo 5 se trata el tema de la seguridad en las máquinas virtuales, sus particularidades y nuevos aspectos

que deben tenerse en cuenta. En el capítulo 6 se analizan vulnerabilidades y ataques que surgieron con la virtualización. En el capítulo 7 se describen las recomendaciones del NIST para la virtualización, se presentan las recomendaciones de seguridad de una nueva norma ISO/IEC y se comparan ambos trabajos. En el capítulo 8 se hace un resumen de lo visto. Por último, en el capítulo 9, se dan las conclusiones de este trabajo.

2 Virtualización

2.1 Antecedentes e historia

Como antecedente a la virtualización de un servidor completo, se puede mencionar la virtualización de la memoria. La computadora Atlas, del año 1959, fue la primera en usar memoria virtual [14]. Esta tecnología permitía a las aplicaciones usar más memoria que la existente en un equipo. Simulaba mayor tamaño gracias a que usaba, además de la memoria existente, el disco rígido para almacenar la información que se suponía estaba en la memoria principal. La Unidad de gestión de memoria[4] junto con el sistema operativo son los responsables de traducir las direcciones de memoria virtual solicitadas por las aplicaciones a direcciones de memoria real (mediante el uso de *page table*[5]). El sistema operativo, a través del supervisor de páginas (*page supervisor*), también se ocupa de cargar en memoria real los bloques de memoria guardados en el disco rígido al momento de ser requeridos.

El comienzo del uso de la virtualización de *hardware* se remonta a la década de 1960 [1, 4, 15], cuando se usó para dividir recursos de las computadoras *mainframe* de IBM (el primer término utilizado para referirse a este concepto era el de *timeshare*, tiempo compartido). Mediante un hipervisor, que llamaron CP (por sus siglas en inglés, *Control Program*), se creaban múltiples MVs independientes. Cada una de ellas simulaba ser un *hardware* donde corría su sistema operativo *Cambridge Monitor System* (CMS, por sus siglas en inglés, que luego se denominó *Conversational Monitor System*). En las primeras versiones del CP no se había implementado un

[4] La Unidad de gestión de memoria, comúnmente llamada MMU, por las siglas en inglés de *Memory Management Unit*, es un dispositivo que maneja los accesos a la memoria del procesador. Entre sus responsabilidades se encuentra la de traducir las direcciones lógicas en físicas, proteger el acceso a la memoria y controlar la cache.

[5] La *page table* (en castellano tabla de paginación) es una estructura de datos donde se almacenan direcciones virtuales con sus respectivas direcciones físicas o reales. Las direcciones virtuales son las que manejan los procesos y las físicas son las de la memoria RAM. Cuando se necesita hacer una traducción de dirección virtual a física, la MMU, antes de acceder a la *page table*, consulta en una *cache* de esta tabla, llamada *Translation Lookaside Buffer* (TLB, en castellano *buffer* de traducción adelantada). Si la dirección buscada se encuentra en esta *cache* no es necesario acceder a la *page table*.

correcto aislamiento entre las distintas MV ni técnicas para segmentar la memoria, generando así un problema de seguridad [16]. Lo que se buscaba era optimizar el uso de recursos y reducir su costo de adquisición y mantenimiento.

En 1974 Popek y Golberg [17] introdujeron los requerimientos para la virtualización. Mediante técnicas formales definieron condiciones suficientes (aunque no necesarias) para verificar si una arquitectura soporta MVs de manera eficiente. Según ellos, un gestor de MVs tiene tres características esenciales. La primera es la de equivalencia. Esta propiedad indica que un programa que corra en un ambiente virtual debe tener un comportamiento y un resultado similar a otro que corra en una máquina real. La segunda se refiere a la eficiencia. Señala que los programas que corren en el ambiente facilitado por el gestor de máquinas virtuales muestran, en el peor de los casos, una mínima disminución en la velocidad de ejecución (la mayoría de las instrucciones del procesador virtual se deben ejecutar directamente por el real, sin participación del gestor de MVs). La tercera es la de control de recursos. Indica que el gestor de MVs debe tener el control total de los recursos del sistema (memoria, procesador y periféricos). Un programa en un ambiente virtual solo puede acceder a los recursos que explícitamente le han asignado y el gestor puede recuperar estos recursos en caso de requerirlo.

De los requerimientos de Popek et al. [17] los procesadores con arquitectura x86[6] cumplían con el de equivalencia y el de control de recursos, pero no con el de eficiencia [18]. Algunas instrucciones ejecutadas en la MV (las que podían interferir con el estado del gestor de MVs o el SO anfitrión) debían traducirse en varias instrucciones en el procesador real, lo que generaba una sobrecarga que afectaba la eficiencia [19]. En 2005 Intel incorporó una extensión llamada VT-x [20] que agregó en sus procesadores soporte de *hardware* para brindar de manera eficiente la virtualización. En 2006 AMD incorporó su propia extensión a sus procesadores, llamada AMD-v [21], para dar un soporte bastante similar al de Intel [18]. Estas mejoras

[6] x86 es una familia de procesadores compatibles con el conjunto de instrucciones del procesador Intel 8086.

permitieron a los procesadores x86 cumplir con los requisitos de Popek y Goldberg [18].

2.2 Definición

La virtualización es la creación de varios recursos lógicos a partir de un único recurso físico. Con un solo recurso (por ejemplo, una computadora completa, un dispositivo de almacenamiento, una red o un servicio) se crean múltiples recursos simulados (varias computadoras, dispositivos de almacenamiento, redes o servicios). La virtualización se realiza mediante programas que simulan la funcionalidad de los recursos virtualizados. Un ejemplo de virtualización es una MV, que simula ser un equipo real autónomo, con todos sus recursos de *hardware* y su propio sistema operativo. Además de *hardware* se pueden virtualizar programas, aplicaciones, servicios o sesiones de usuario.

Existen dos tipos de virtualización de servidores. Uno es la virtualización completa, donde el sistema operativo invitado no sabe que corre sobre un equipo virtual. A este tipo se la denomina virtualización completa (*full virtualization* en inglés). El segundo tipo es llamado paravirtualización (*paravirtualization* en inglés). En este tipo de virtualización, no se emula todo el *hardware* subyacente, sino que se genera una interfaz especial para que el sistema operativo invitado pueda colaborar con el hipervisor [22]. En esta interfaz se eliminan instrucciones y se agregan otras. A las instrucciones que se agregan se las denominan *hypercalls*[7] [19]. De la misma manera que una aplicación hace una llamada al sistema (en inglés *system call*) para requerir un servicio al sistema operativo, las *hypercalls* son usadas por las MVs para ejecutar instrucciones privilegiadas [23]. El código fuente del *kernel* del sistema operativo invitado debe sufrir modificaciones para generar *hypercalls* en lugar *de system calls*. Esto se debe a que en la paravirtualización el sistema operativo se altera para poder ejecutarse en la capa de privilegios de nivel 1, dejando al nivel 0, nivel más privilegiado, para una ejecución segura del hipervisor. Esto supone una mejora en la performance de los sistemas

[7] El origen del nombre surge de que estas *hypercalls*, llamadas al hipervisor (gestor de MVs), cumplen una función similar a las *system calls*, llamadas a sistema.

paravirtualizados, en comparación con los sistemas de virtualización completa, ya que estos precisan realizar costosas traducciones binarias. Debido a las modificaciones que introduce esta nueva interfaz, el sistema operativo invitado es consciente de que no está corriendo sobre una máquina real.

En la siguiente figura se muestra un ejemplo de virtualización de dispositivos de *hardware*:

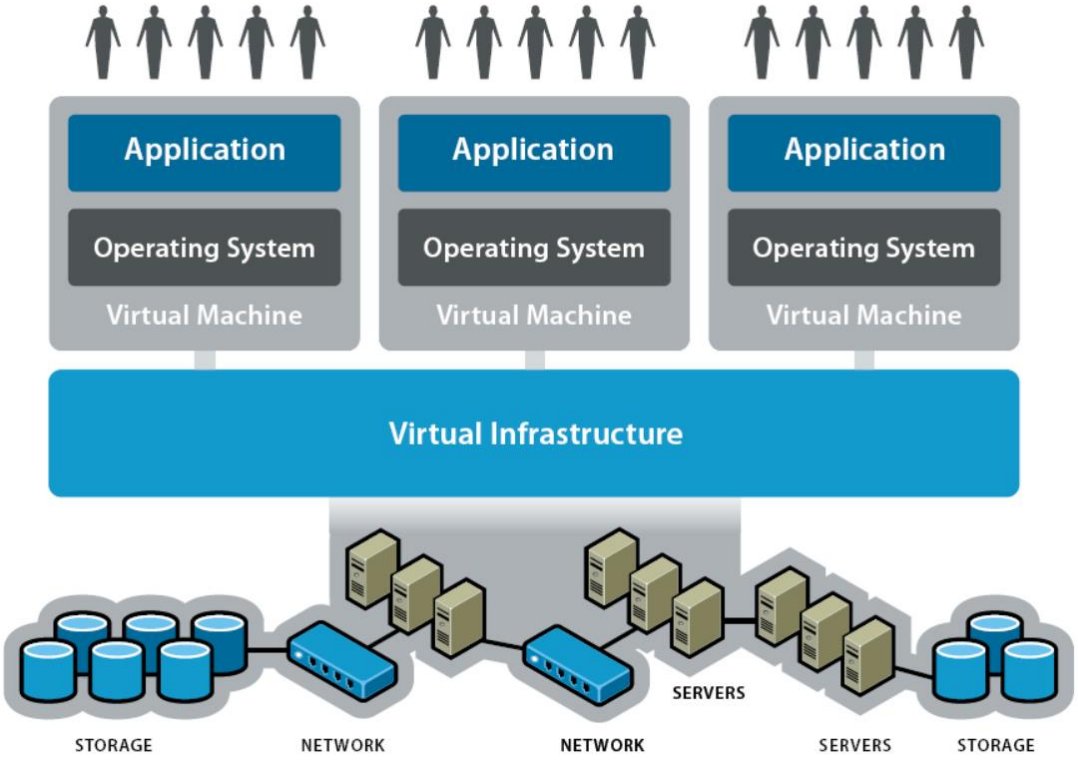

Figura 1. Ejemplo de infraestructura virtual que da soporte a tres servidores. Fuente: [24]

2.3 Hipervisor

Es la capa de *software* que permite la virtualización. Provee una plataforma virtual sobre la cual estarán los sistemas operativos invitados llamados MVs [11]. Es responsable de crear, correr, administrar y controlar las MVs, que pueden ejecutarse directamente sobre el *hardware* o bien sobre un sistema operativo. También llamado *Virtual Machine Monitor* (VMM, por sus siglas en inglés), es el que crea una capa de abstracción entre el *hardware* de la máquina física (*host*) y el sistema operativo de la MV (sistema operativo

invitado). Permite crear una o varias instancias de ordenadores virtuales que se ejecutan en un mismo ordenador físico.

Tradicionalmente se llamó al *kernel* del sistema operativo *el supervisor*. De ahí deriva el término hipervisor, que sería el supervisor del supervisor [14]. Se suele señalar que este término fue acuñado en 1972 cuando IBM actualizó su gestor de MVs, CP, para su *mainframe* System/370 [25, 26], sin embargo, se usaba, por lo menos, desde 4 años antes[8].

Esta capa de *software* maneja, gestiona y arbitra los cuatro recursos principales de una computadora (procesador, memoria, almacenamiento y conexiones de red) lo que le permite repartir dinámicamente dichos recursos entre todas las MVs definidas en el computador central. Si bien todas las MVs que corren en un mismo ordenador son independientes entre sí (cada una cree que tiene su propio procesador, memoria, etc.), físicamente están compartiendo los recursos de *hardware* subyacentes.

Actualmente los fabricantes de procesadores Intel y AMD dan soporte de *hardware* a la virtualización. Para ello Intel introdujo a fines de 2005 su tecnología Intel-VTx y a principios de 2006 lo hizo AMD, con AMD-V[9] [27]. Estas extensiones pueden acelerar la virtualización si son aprovechados por los hipervisores. Por ejemplo, pueden traducir direcciones de memoria virtual a direcciones de memoria física.

2.3.1 Tipos de hipervisores

La virtualización de servidores se puede realizar en distintas capas. Una opción es directamente sobre el *hardware*, la otra es sobre un sistema operativo, como cualquier otra aplicación.

- **Tipo 1, *bare metal* o nativo**: se ejecuta directamente sobre el *hardware*. Permite gestionar sistemas operativos invitados que se

[8] En 1970 aparece en dos *papers*, "Operating systems architecture" de H Katzan Jr y en "Analysis of Major Computer Operating Systems" de Clinton S. McIntosh, Kenneth P. Choate y William C. Mittwede.

[9] Los primeros procesadores que incorporaron la tecnología de Intel fueron los modelos 662 y 672 del Pentium 4. En el caso de AMD los procesadores que inauguraron este soporte fueron el Athlon 64 ("Orleans"), el Athlon 64 X2 ("Windsor") y el Athlon 64 FX ("Windsor").

ejecutan en otro nivel por encima del hipervisor. El término *bare metal* (metal desnudo o pelado) hace referencia a la parte física del servidor, al *hardware*. Es una expresión metafórica que señala que algo ocurre a muy bajo nivel, en este caso, directamente sobre el *hardware*.

- **Tipo 2 o virtualización de *host***: se ejecuta sobre un sistema operativo, como cualquier otra aplicación. Con la capa del hipervisor como un segundo nivel de *software* distinta, se pueden ejecutar distintos sistemas operativos invitados en el tercer nivel por encima del *hardware*.

En las siguientes imágenes se representan ambos tipos.

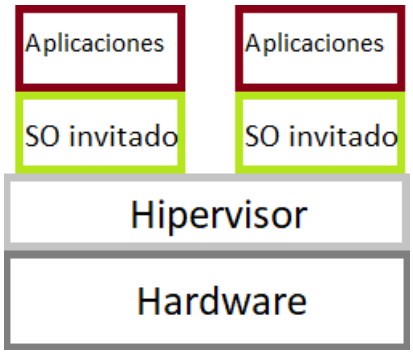

Figura 2. Hipervisor de tipo 1, bare metal o nativo

Figura 3. Hipervisor de tipo 2 o de host

Ejemplos de hipervisor de tipo 1 son Xen Hypervisor, Microsoft Hyper-V y VMware ESXi. Del Tipo 2 algunos ejemplos son VMware Fusion, Oracle Virtual Box, Solaris Zones, Parallels, Windows Virtual PC y QEMU.

Al tener acceso directo al *hardware*, el hipervisor de tipo 1 es más eficiente que la segunda alternativa [24, 28] y aprovecha mejor el soporte de *hardware* a la virtualización [14]. La ISO 21878 [29] recomienda el uso de hipervisores de tipo 1 ya que reduce la superficie de ataque. Por estos motivos, en general, el de tipo 2 es usado en aplicaciones menos críticas [11].

Como un hipervisor es mucho más simple y chico que un sistema operativo [7], como objetivo de ataque es más pequeño. Reemplazar un sistema operativo con un hipervisor, como se usa en virtualización de tipo 1, puede

aumentar la seguridad [7], siempre que esté bien asegurado. Agregar un hipervisor sobre el sistema operativo del *host*, como ocurre en virtualización de tipo 2, tiende a incrementar los riesgos de seguridad [7].

3 Ventajas y desventajas de la virtualización

Son múltiples las ventajas de trabajar con un ambiente virtualizado. Las principales se consideran que son el aislamiento y la maximización en el aprovechamiento de los recursos [11]. A continuación, se detallan las ventajas.

- **Optimización en el uso del *hardware***. Se pueden combinar servidores que tengan mayor carga de trabajo en distintos horarios. Además, mediante el uso de MVs, se pueden mover servidores virtualizados entre distinto *hardware* de manera transparente para los usuarios de esos servidores, según las necesidades de recursos.
- **Reducción de tiempos**. Mejora en los tiempos de creación de ambientes de desarrollo, pruebas, capacitación y producción. A partir de *templates* de sistemas preconfigurados de manera estandarizada según las necesidades (a estos *templates* se los denomina *Golden Images*), se puede replicar ambientes de manera rápida y sencilla. También hay una mejora en los tiempos al migrar una MVs de un equipo físico a otro, al compararlo con una migración de un servidor físico.
- **Facilita la continuidad del negocio**. Gracias a que servidores enteros virtuales se pueden encapsular en un solo archivo se facilita el *backup*, restauración y réplicas de los servidores. Se reduce así la complejidad y los costos de la recuperación ante desastres y de la continuidad del negocio.
- **Mayor flexibilidad**. Permite correr múltiples instancias de uno o varios sistemas operativos en un solo equipo físico. Facilita la escalabilidad [30] al simplificar las tareas para ampliar o reducir los recursos de *hardware* (procesador, memoria, etc.) según cambien los requerimientos de los sistemas.
- **Uso de sistemas heredados**. Facilita el uso de aplicaciones antiguas, que corren sobre sistemas operativos obsoletos (sin soporte técnico de sus fabricantes). Esto se logra gracias a controles que no se pueden aplicar de manera directa en esos sistemas, pero que sí se pueden realizar en el gestor de la MV o en el sistema operativo del equipo

anfitrión [7]. Ejemplos de estos controles son políticas de autenticación, análisis de tráfico o comportamiento y herramientas de auditoría.

- **Costos**. Al reducir la cantidad de servidores genera un ahorro en la electricidad requerida. Además, al reducir los servidores, se necesita un menor espacio físico para el centro de datos. Esto también genera una menor demanda de refrigeración. Al simplificar las tareas de operación también se reducen los costos [3].

Desde el punto de vista de seguridad, la virtualización tiene varias ventajas que se describen a continuación:

- **Análisis forense**. Si en un ataque se compromete una máquina virtual, esta se puede clonar y realizar un análisis forense en un ambiente no productivo. De esta manera se simplifica el análisis del equipo atacado, ya que se puede analizar sin interferir con la recuperación del equipo afectado. Si se trabaja con un *snapshot*[10] el análisis forense cuenta con la memoria completa, algo que es muy difícil de conseguir si se trabaja sobre un equipo físico. Esto se debe a que la recolección de la memoria en los equipos físicos no siempre es posible (ya sea por falta de herramientas, falta de acceso a donde está el equipo, etc.) y, si se pudiese realizar, la acción de recolectarla suele alterar la memoria y puede llegar a corromperla sobrescribiendo datos relevantes para la investigación[11]. Otra ventaja de trabajar con *snapshot* es que el analista forense podría hacer pruebas sobre la máquina virtual en funcionamiento, modificar datos, etc. sin alterar ninguna información de la máquina original [31].
- **Rápida recuperación ante ataques**. Una máquina comprometida puede reemplazarse inmediatamente por una imagen anterior sin el problema generado por el ataque. Esto se debe gracias a la movilidad

[10] Un *snapshot* es una captura de todo el estado de una MV en un determinado momento. Incluye sus discos, memorias, otros dispositivos, su estado (si está prendida, apagada), etc.

[11] Si bien, a veces, se podría forzar un volcado de memoria (*memory dump* en inlgés) para obtener la memoria completa del equipo a analizar, no siempre es posible (por restricciones del *hardware* o del sistema operativo, porque, en algunos casos, conseguir el volcado implica un apagado del equipo y no siempre esto se puede realizar, etc.

que permiten las MVs, que fácilmente pueden copiarse y transportarse a otro lugar. De esta manera, los *backups* de las imágenes de las MVs pueden realizarse con poco esfuerzo [32].

- **Emparchar es más seguro**. Antes de actualizar, se puede clonar fácilmente un servidor de producción, aplicar el parche en el clon, verificar que todo siga funcionando correctamente y luego reemplazar con este clon la imagen productiva. Otra alternativa es realizar una copia de la imagen productiva antes de actualizar. Esta copia queda como respaldo. Si al instalar el parche surge algún problema en producción, rápidamente se puede restaurar la versión sin actualizar.
- **Análisis de programas sospechosos**. Permite generar ambientes para analizar el comportamiento de aplicaciones que pueden ser maliciosas. El uso más común de un ambiente virtualizado, para un investigador de código malicioso[12], es ejecutar el código a analizar y observar cómo se comporta. Al trabajar sobre un ambiente virtualizado, una vez que finaliza el análisis se puede eliminar el ambiente sin riesgos sobre el sistema real que lo hospedaba [33].

Algunos autores presentan distintos escenarios en los cuales con la virtualización se obtienen otros beneficios. Un ejemplo interesante es el de Chen et al. [34], que proponen que los sistemas operativos y aplicaciones actualmente existentes se migren a MVs. Esta estructura permite habilitar servicios por debajo del sistema operativo. Los servicios que da como ejemplo son el de *logueo*, el de prevención y detección de intrusiones y el de migración de ambientes. Afirma que asegurar el gestor de virtualización es más fácil al compararlo con un sistema operativo (debido a que es más sencillo y más chico). Al mover estos servicios por debajo del sistema operativo, estos pueden seguir funcionando incluso si el sistema operativo que corre sobre la MV es comprometido.

La desventaja de esta tecnología es la sobrecarga que genera para realizar la virtualización [35]. Esta sobrecarga varía según el tipo de

[12] El código malicioso (o *malware* en inglés) es un programa creado con el objetivo de dañar o infiltrarse en un sistema informático, produciendo una violación de una política de seguridad.

virtualización usado, de las herramientas utilizadas, de los recursos virtualizados y de las instrucciones del procesador usadas. Desde el punto de vista de seguridad, presenta aspectos que se deben considerar y tener en cuenta, que se verán más adelante.

4 Tipos de virtualización

Hay muchos posibles tipos de virtualización, entre los más importantes se destacan el de servidores, el de escritorio, el de aplicación, el de red, el de almacenamiento y el de servicios.

4.1 Virtualización de servidores

La virtualización de servidores, también llamada de *hardware*, crea una capa abstracta entre el *software* y el *hardware* subyacente. Para ello se instala un hipervisor (también llamado administrador de MVs [36, 29]) que maneja esta nueva capa, dentro de la cual estarán los servidores o MVs. Cada MV tendrá su propio sistema operativo, que corre de manera aislada y en simultáneo con otros servidores virtuales. A la MV se la suele llamar invitada o huésped (*guest* en inglés) y a la máquina física anfitrión o *host* en inglés. Ejemplos de productos que implementan esta tecnología son VMware, Hyper-V de Microsoft, Xen, KVM, VirtualBox de Oracle y QEMU, entre otros.

Entre sus ventajas se puede destacar:

1. Reducción de costos [3].
2. Gestión simplificada de infraestructura, lo que reduce tiempos [3].
3. Mayor disponibilidad de servicios. Se puede realizar mantenimiento del *hardware* sin necesidad de parar la MV [37]. Otra característica que aumenta la disponibilidad es la migración de una máquina virtual en funcionamiento de un equipo físico a otro, ya que facilita la continuidad de los servicios en caso de algún problema en el equipo físico donde está corriendo [3].
4. Independencia de sistemas *legacy.*

4.2 Virtualización de escritorio

La virtualización de escritorio (llamada VDI, por las siglas en inglés de *virtual desktop infrastructure*) permite la separación entre el escritorio que

usan los usuarios (cuyos datos, programas y sistema operativo residen en un servidor central) y la máquina física que el usuario utiliza para acceder [38]. El escritorio virtualizado se guarda remotamente en un servidor central en lugar de en el disco duro del ordenador personal. Esto significa que cuando los usuarios trabajan en su escritorio desde su portátil u ordenador personal, todos sus programas, aplicaciones, procesos y datos se almacenan y ejecutan centralmente, permitiendo a los usuarios acceder remotamente a sus escritorios desde cualquier dispositivo capaz de conectarse remotamente al escritorio, tales como un portátil, una computadora personal, un teléfono inteligente o algún otro cliente ligero.

Esta virtualización reduce la dependencia del dispositivo usado, que se puede reemplazar en caso de algún problema. Sin embargo, la VDI genera una dependencia completa de la conectividad. Si hay degradación o interrupción del servicio de red se genera una degradación o interrupción en el servicio.

Ofrece las siguientes ventajas:

1. Independencia de sistemas *legacy*
2. Reducción de costos: instalar, reparar y mantener las aplicaciones se hace desde una ubicación central. Uso más eficaz de la infraestructura.
3. Reducción de los tiempos para el alta de nuevos sistemas.

4.3 Virtualización de aplicación

La virtualización de aplicación encapsula los programas informáticos separándolos del sistema operativo en que se ejecutan. De esta manera se logra que un usuario pueda usar aplicaciones que corren en diferentes sistemas operativos y distintas plataformas de *hardware*.

Las aplicaciones no se instalan directamente en la computadora, sino que se instalan con los recursos que necesitan para correr. Entre estos recursos están las interfaces usadas por la aplicación, que no interactúa directamente

con el sistema operativo. Un ejemplo son las aplicaciones hechas en Java[13], que no se ejecutan directamente sobre el sistema operativo, sino que lo hacen sobre un entorno virtual provisto por la máquina virtual de Java (JVM, por las siglas en inglés de *Java Virtual Machine*) [7]. Otros ejemplos de tecnologías que la implementan son: Citrix y VMware

Las ventajas que ofrece son:

1. Las aplicaciones que son incompatibles entre sí pueden trabajar juntas con la virtualización.
2. Aplicaciones incompatibles con nuevas versiones de un sistema operativo pueden seguir siendo utilizadas.
3. Varias instancias de una aplicación pueden iniciarse automáticamente en otras máquinas, cuando hay demasiada carga de trabajo para la misma.
4. Mejora en la disponibilidad: si falla alguna función crítica se puede conmutar de un servidor a otro.
5. Reducción de costos: Dado que la instalación, actualización y administración de aplicaciones se hace desde una ubicación central, las mismas pueden ser fácilmente copiadas a sistemas remotos.

4.5 Virtualización de red

La virtualización de red presenta una capa de red virtual, a partir de la combinación de componentes de red reales (*hardware* y *software*), que oculta la red física mientras brinda un servicio similar. Busca facilitar el uso compartido de recursos e independizar los componentes físicos usados de los ofrecidos como servicio. Este enfoque de virtualización oculta la red física a los clientes y servidores de la red.

Ofrece las siguientes ventajas:

[13] Java es un lenguaje de programación. Originalmente fue desarrollado por la empresa Sun Microsystems, que posteriormente fue adquirida por la empresa Oracle. Una de las características de este lenguaje es que un mismo código compilado se puede ejecutar sobre máquinas virtuales de Java en distintos sistemas operativos, sin necesidad de recompilarlo.

1. Aumenta el rendimiento y hace un uso más eficiente de los recursos.
2. Facilita la seguridad de datos al aislar la red interna de todo el tráfico de redes externas en uso. Sólo el tráfico autorizado de la red se le permite entrar y salir de la red del centro de datos.
3. Alta disponibilidad: las organizaciones pueden instalar distintas conexiones de red de diferentes proveedores. Si alguno de estos vínculos falla, la función de virtualización es redirigir los mensajes a una de las conexiones de red sobrevivientes.
4. Ahorro de recursos físicos: algunas industrias reguladas requieren que la información personal o confidencial se mantenga en sistemas y redes separadas. La virtualización de la red hace posible segmentar una red única en múltiples redes independientes sin necesidad de que la organización instale cableado independiente y nuevos equipos de red.

4.6 Virtualización de almacenamiento

Se refiere al proceso de separar el almacenamiento lógico del almacenamiento físico, logrando la independencia de la ubicación mediante la abstracción de la localización física de los datos. Oculta los dispositivos donde las aplicaciones almacenan sus datos, presentándoles un bloque monolítico en el cual trabajar, independientemente de la cantidad de dispositivos físicos usados y de su localización. Facilita que distintas aplicaciones, corriendo en un mismo servidor o en distintos servidores, pueden compartir los datos de manera transparente [14].

Esta virtualización puede, dependiendo de su implementación, mejorar el desempeño gracias al uso de *caches* y la combinación de discos de estado sólido y disco duros [3].

Además, facilita la migración de datos, y su resguardo, de manera concurrente a los accesos que pueden realizar las aplicaciones. Otra ventaja es que permite ampliar la cantidad de espacio de almacenamiento disponible para las aplicaciones de manera transparente para ellas, a medida que

cambien sus requerimientos. Este almacenamiento centralizado y virtualizado, reduce los problemas de acceso, reduce los costos y mejora la eficiencia en el manejo de la información [14].

4.7 Virtualización de servicios

La virtualización de servicios es una técnica que emula el comportamiento de un componente de *software*, por ejemplo, un *web service*[14] o una base de datos. Permite que una aplicación consuma un servicio virtual en vez del servicio real. Son múltiples los escenarios donde es muy útil esta virtualización, por ejemplo, cuando se quiere probar un componente que depende de otros servicios, pasibles de virtualización, y que por algún motivo no están disponibles (porque aún no se han desarrollado, porque han dejado de funcionar, porque tienen un costo por cada vez que se utilizan, etc.). El uso de esta técnica en ambientes de desarrollo y pruebas reduce la dependencia entre los distintos equipos, ya que cada uno puede trabajar en su producto usando servicios virtualizados de los otros equipos. Los beneficios que generan los servicios virtualizados reducen costo y tiempo.

Existen varios productos para este tipo de virtualización, por ejemplo, *ServiceV Pro* y *Soap UI* de *SmartBear, CA Service Virtualization* de *ITKO*, *IBM Rational Test Virtualization* y *HP Service Virtualization.* Estos productos permiten de manera muy sencilla y rápida virtualizar servicios. Una opción que ofrecen es, a partir de un *WSDL*[15], definir automáticamente las interfaces de los servicios disponibles. Una vez que se han agregado los servicios se debe definir qué deben devolver al ser consumidos. Una alternativa es establecer un valor fijo o se puede definir reglas para que, dependiendo de los parámetros recibos, devuelva distintos valores.

[14] Un *web service* es un servicio que está disponible en una red informática. Para acceder y consumir estos servicios se usa el protocolo de comunicación HTTP.

[15] Un *WSDL* (por las siglas en inglés de *Web Services Description Language*) es un archivo xml, con un formato estándar, que permite describir servicios publicados en un *web server*. En él se explicita el nombre de los servicios, el tipo de los parámetros que reciben y el tipo del valor que devuelve, entre otra información.

Suelen ofrecer un mínimo de tres modos de funcionamiento. Uno es el “modo puente”, en el cual, al ser invocado un servicio en el virtualizador, este llama al servicio real y se devuelve su respuesta. Otro es el “modo aprendizaje”. Es similar al anterior, pero guarda la información aprendida (ahora sabe que si se consume un servicio con determinados parámetros se debe devolver el valor devuelto por el servicio real). Por último, existe el “modo virtualización” en el cual yo no se invoca más al servicio real, sino que se devuelve lo definido, lo aprendido o algún valor por defecto en caso de que se consuma el servicio con parámetros no aprendidos.

En escenarios de desarrollo y pruebas se puede hacer que todos los servicios sean publicados en la herramienta de virtualización, incluso antes de que se hayan desarrollado, para que puedan ser usados de manera inmediata por los otros componentes que dependen de él. Una vez que los servicios se vayan desarrollando, la aplicación de virtualización se configura para que apunte al servicio real y aprenda su comportamiento por un determinado tiempo. Luego se pone el servicio en modo puente y se deja en ese modo. Si en algún momento este servicio presenta algún problema, automáticamente se cambia al modo virtualización y todos los demás, que dependen de él, pueden seguir funcionando hasta que se solucione el inconveniente.

4.8 Virtualización de sistema operativo

La virtualización de sistema operativo (VSO) genera múltiples instancias aisladas de espacios de usuario. Estos espacios aislados se denominan contenedores, en los cuales se pueden separar las aplicaciones, para que no se vean entre sí. La VSO es usada para facilitar el despliegue de aplicaciones en distintos ambientes, ya que permite empaquetar las aplicaciones con sus dependencias. Una implementación actualmente muy difundida y apreciada de este tipo de virtualización es Docker [12].

En la VSO se genera una nueva capa de abstracción que aísla la vista que tienen las aplicaciones sobre el sistema operativo subyacente. En un gestor de MVs usado en la virtualización de servidores, la capa generada presenta

el *hardware* virtual. En cambio, en la VSO, la nueva capa ofrece espacios de usuario aislados, como se observa en la Figura 4.

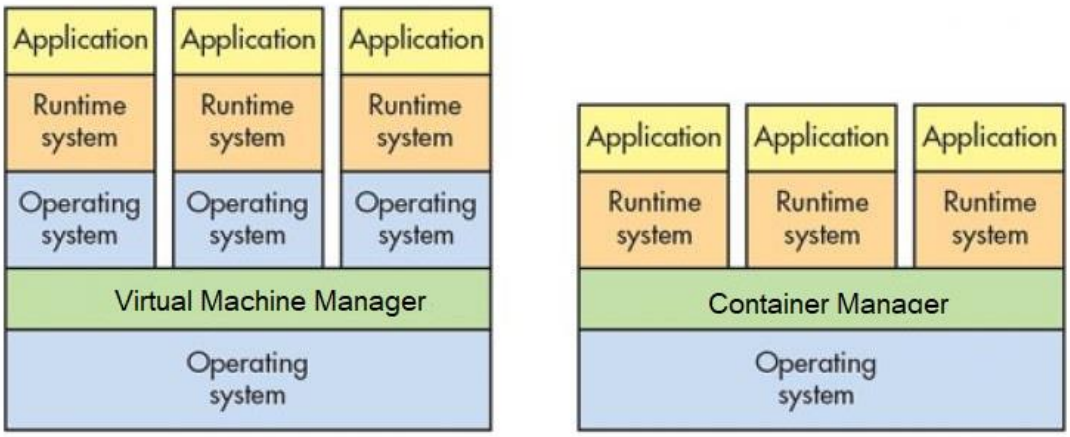

Figura 4. Virtualización de servidores vs sistema operativo. Fuente [39]

La VSO no tiene la flexibilidad de un virtualizador de servidores, pero tiene la ventaja de que normalmente es más liviano y no requiere los recursos que sí necesita un virtualizador de servidores. Este último debe simular el procesador, la memoria, el almacenamiento, la red, etc. y soportar un sistema operativo. El VSO no realiza todo este trabajo y genera menos sobrecarga. Por ello se pueden crear nuevos contenedores muy rápidamente [39].

La VSO no permite hospedar un sistema operativo distinto al del *host*, algo que sí se puede hacer en virtualización de *hardware*. En la VSO todos los contenedores comparten un mismo sistema operativo y cada contenedor da soporte a aplicaciones, no a sistemas operativos. En la virtualización de *hardware* las MVs tienen cada una su propio sistema operativo (que pueden ser distintos entre sí) y comparten entre ellas el *hardware* subyacente.

5 Seguridad en máquinas virtuales

Previamente se han mencionado importantes beneficios que ofrece la virtualización. Sin embargo, adoptar estas tecnologías requiere cuidados que deben tenerse en cuenta. La virtualización crea nuevos desafíos, se incrementa la posibilidad de vulnerabilidades, se requieren personas capacitadas en la nueva tecnología y se agrega complejidad en la administración de los sistemas. En comparación con las máquinas físicas, crear MVs es muy sencillo. Esto ha generado un gran crecimiento de su uso, sin los controles de seguridad necesarios. A esta problemática generada por la proliferación de MVs, sin los controles y monitores adecuados, se la denomina *VM sprawl* [7]. En 2008, según el analista de *Gartner* Neil McDonald [40], más del 60% de las MVs en producción eran menos seguras que las máquinas físicas. Además, el hecho de que varios sistemas antes separados ahora se encuentren en un mismo equipo, puede causar un impacto mayor si se compromete la seguridad del equipo *host* [7]. A continuación, se mencionan distintos problemas comunes en virtualización.

Los dos hipervisores más usados son los de VMware e Hyper-V [28, 41]. Según un estudio de Tsai [41] entre los dos tienen un 94% del mercado en 2018. A medida que crecía el uso de la virtualización y el interés por estos productos aumentaba, también lo hacía la cantidad de vulnerabilidades detectadas en ellos, como se observa en la tabla 1, generada a partir de las vulnerabilidades publicadas en CVE[16].

[16] CVE son las siglas en inglés de *Common Vulnerabilities and Exposures*. Allí se registra una lista pública de vulnerabilidades de seguridad conocidas. Su objetivo es ofrecer una nomenclatura estándar para el conocimiento público de estos problemas. Fue creado y es mantenido por *The MITRE Corporation*. Su sitio oficial es https://cve.mitre.org/

Año	VMWare	Xen
1999	1	
2000	1	
2001	1	
2002	1	
2003	4	
2004	4	
2005	8	
2006	6	
2007	25	2
2008	31	2
2009	20	2
2010	24	-
2011	18	-
2012	34	35
2013	18	42
2014	17	45
2015	15	41
2016	36	28
2017	45	62
2018	16	21
Total	325	280

Tabla 1: *CVE* por año por producto

En el Anexo A se muestra en detalle todos los errores reportados, para estos productos, en CVE, por año, separados por tipo de vulnerabilidades.

Una de las características fundamentales de las MVs, el aislamiento que estas tienen con otras MVs y con el sistema operativo anfitrión (si el hipervisor es de tipo 2), ha sido vulnerada en reiteradas oportunidades, por lo menos, desde 2008[17]. Ese año, dos investigadores argentinos, Gerardo Richarte y

[17] Las vulnerabilidades previamente detectadas generaban, principalmente, errores de tipo *Stack Overflow* (ocurre cuando una función copia datos en un bloque de memoria que no

Nicolás Economou, de *Core Security Technologies*[18] publicaron una vulnerabilidad[19] (CVE-2008-0923) que afectaba a distintos productos de VMware[20] y permitía a los usuarios del sistema operativo invitado leer y escribir en el sistema de archivos del equipo *host* (incluida la carpeta de sistemas y otros archivos sensibles en cuanto a seguridad) [42]. A este tipo de ataque, donde desde una MV se accede al sistema operativo del *host*, se lo denomina **escape de la MV** (o *jail break*, fuga de la cárcel en castellano). Otro ejemplo de este tipo de ataque lo presentó Kortchinsky en *Black Hat*[21] en 2009 [43]. Allí utilizan el adaptador virtual de pantalla PCI "VMware SVGA II" para lograr una fuga de información del *host* (el invitado puede leer parte de la memoria del *host*). Además, puede, desde el sistema operativo invitado, escribir en la memoria del *host*, lo que le permite ejecutar código en el servidor físico.

Para Randel [44], la mayoría de las cuestiones de seguridad no tienen que ver con la infraestructura de virtualización en sí misma sino con los asuntos operacionales. Pasar los recursos de sistemas a un ambiente virtualizado tiene poca o ninguna consecuencia en cuanto a las amenazas y vulnerabilidades de los activos [7]. Las vulnerabilidades de un programa o servicio se mantienen cuando se pasa a un sistema virtual, aunque ayuda a que el impacto por la explotación de la vulnerabilidad en el ambiente virtual sea menor que antes, lógicamente dependiendo de la implementación que se haga.

---

estaba asignado para ella), *Denegación de Servicio* (el ataque consume demasiados recursos generando falta de disponibilidad), *Ejecución Arbitraria de Código* (permite ejecutar comandos inicialmente no permitidos) o *Ganar Privilegios* (permite que un usuario o servicio obtenga más privilegios que los inicialmente otorgados).

[18] Empresa fundada en Buenos Aires en 1996, especializada en seguridad informática.

[19] Si bien fue publicada en febrero de 2008, en octubre del año anterior la empresa *Core* le notificó a VMware sobre esta vulnerabilidad.

[20] Workstation 6.0.2, Workstation 5.5.4, Player 2.0.2, Player 1.0.4, ACE 2.0.2 y ACE 1.0.2

[21] Las *Black Hat Briefings* (o simplemente *Black Hat*) son prestigiosas conferencias de seguridad informática que se organizan desde 1997.

Sin embargo, la virtualización puede generar nuevos vectores de ataques [7]. Según Gkortzis [35], se amplía la superficie en la cual se pueden producir ataques, ya que esta tecnología provee acceso a varios servicios a un mismo recurso de *hardware*. Además, los componentes de *hardware* en los cuales ocurre la virtualización (la *cache*[22] del *CPU*[23], la *GPU*[24], etc.) no han sido originalmente diseñados teniendo en cuenta el aislamiento [45]. Esto ha generado vulnerabilidades en su uso compartido que pueden, potencialmente, ser explotadas [46].

Según Ritter [47] y Dhawale [48] en una arquitectura de virtualización se presentan, fundamentalmente, cuatro nuevas características relacionadas con la seguridad. Estas son: una nueva capa, concentración, estado y movilidad. La **nueva capa** es el hipervisor. Como este maneja todas las MVs que corren sobre él, tiene derechos administrativos sobre los componentes virtualizados. Si algún atacante toma el control del hipervisor, puede comprometer todas las MVs que contiene con privilegios elevados. Según Randel [44], esta nueva capa debe ser tenida en cuenta por los nuevos vectores de ataque que se agregan a los conocidos en la infraestructura física. **Concentración** hace referencia al conjunto de MVs que corren sobre la misma máquina física. Esta característica, donde varias e independientes MVs comparten la misma infraestructura física, en inglés, se la denomina *co-resident* (corresidencia)[25]. Un atacante podría tomar el control de la máquina física, del hipervisor o de otras MVs. El ataque donde se salta de la propia MV a otra virtual se conoce como *Virtual Machine Hyper Jumping* o *Virtual Machine Guest Hopping*. Si el atacante logra, desde una MV, comprometer el hipervisor se lo denomina *hyperjacking*.

[22] La *cache* es un componente usado para almacenar datos recientemente accedidos, con el objetivo de reducir el tiempo de futuros accesos.

[23] Por las siglas en inglés de *Central Processing Unit*, unidad central de procesamiento.

[24] Por las siglas en inglés de *Graphics Processing Unit*, unidad de procesamiento gráfico.

[25] A veces, esta corresidencia es buscada intencionalmente (denominado *co-located*, situados juntos [13]) para poder realizar un ataque, por ejemplo por Ristenpart et al. [10].

Otra de las características señaladas por Ritter [47] es el **estado.** Hace referencia a si la MV está prendida, apagada, suspendida o en otro estado personalizado, mientras el *host* sobre el que se ejecuta está prendido. Esta característica presenta nuevos temas relacionados con la seguridad, ya que, por ejemplo, se podría acceder al estado de las máquinas, modificarlo, etc. Los controles sobre la política de seguridad y la aplicación de parches sobre las MVs deben ser independiente de su estado. Los requerimientos de *loging* y de control de acceso al archivo (o a los archivos) que forman la MV (llamado imagen[26] de la MV) tampoco deben depender de su estado. Se debe evitar accesos no autorizados e implementar mecanismos de control de integridad durante todo el ciclo de vida de las imágenes.

La cuarta característica es la **movilidad,** es la capacidad de las MVs de ser movidas desde una máquina física a otra, tanto dentro de un centro de datos, un centro de *backup* o en la nube. Esta característica presenta nuevos desafíos de seguridad a considerar. Las imágenes que contienen las MVs, además de la información sensible que puede tener cualquier disco físico, tienen el contenido de la memoria RAM[27] del momento en que se generaron, lo que aumenta el riesgo [49]. Esta memoria puede contener en texto plano[28] claves de cuentas de usuario o claves usadas para cifrar información. Por su importancia, las imágenes y los *snapshot* deben ser gestionados de manera tal que se garanticen su confidencialidad e integridad.

[26] Esta imagen encapsula toda la MV, su estado y los archivos usados por el sistema operativo invitado y por las aplicaciones que ahí corren. La imagen puede ser un solo archivo o un directorio. Desde el *host* este archivo o directorio se ve y manipula como cualquier otro archivo.

[27] RAM son las siglas en inglés de *Random Access Memory* (memoria de acceso aleatorio). Es la memoria volátil que usan las computadoras para trabajar. Allí se cargan las instrucciones que ejecuta el procesador, además de contener los datos usados por los programas.

[28] El término *texto plano* se refiere a texto con contenido de caracteres legibles. Se usa para distinguirlo de un *texto cifrado*, en el cual no se puede interpretar el contenido sin tener su clave de descifrado.

6 Vulnerabilidades y ataques en virtualización

Las vulnerabilidades en las distintas implementaciones de MVs crece constantemente (en el Anexo A se presenta una lista de las vulnerabilidades publicadas para Xen y VMware). Aparecen por errores de programación en los productos, por malas configuraciones o por aspectos que no se habían tenido en cuenta durante su diseño o desarrollo. A partir de las vulnerabilidades surgen distintos tipos de ataques. Hay ataques que permiten hacer una denegación de servicio, tomar el control del hipervisor o escapar de la MV. En las siguientes secciones, se presentan distintos tipos de ataques que surgieron con la virtualización.

6.1 Ataque al hipervisor a través del sistema operativo *host*

Este ataque se realiza explotando vulnerabilidades del sistema operativo *host*, en el que corre el hipervisor de tipo 2 [50]. Una vez que el atacante toma el control del sistema operativo, el hipervisor queda comprometido. El atacante puede ahora realizar cualquier actividad maliciosa en las MVs que corren en el hipervisor. Al ataque donde se toma el control del hipervisor se lo llama *hyperjacking* [4]. Esta propagación del ataque se puede ver en la siguiente imagen.

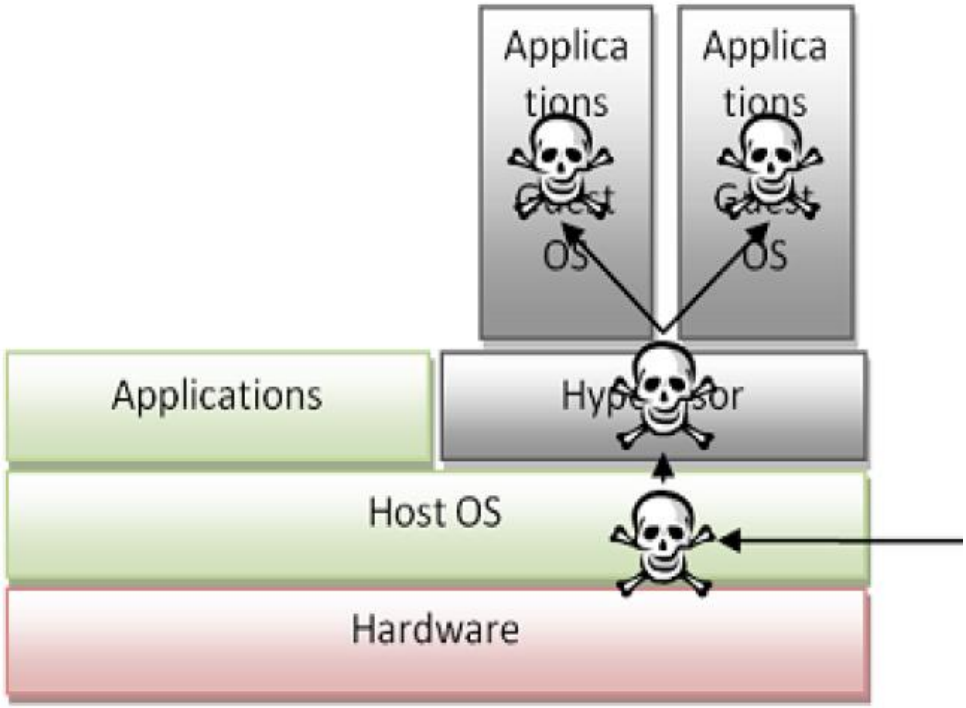

Figura 5. Ataque al hipervisor a partir del sistema operativo host. Fuente: [32]

6.2 Ataque al hipervisor a partir del sistema operativo invitado

Para este ataque se usa el sistema operativo invitado para ganar acceso no autorizado al hipervisor o a otras MVs. A esto se le suele llamar *escape de la MV* o *ataque jailbreak*. Según Dhawale [32] este es un ataque más verosímil que el presentado en el punto anterior, ya que el atacante, en principio, sólo puede comprometer la MV en la que está corriendo la aplicación o el servicio atacado, mientras que el sistema operativo *host*, a priori, le es invisible. Sin embargo, como varias MVs comparten los mismos recursos físicos, si un atacante puede encontrar cómo se mapean los recursos de la MV en la que está corriendo con los físicos, podría atacar directamente a los recursos físicos. Si modifica maliciosamente la memoria virtual puede afectar los recursos físicos compartidos y atacar a las demás MVs, al hipervisor y potencialmente a otros programas en la máquina. Un ejemplo de este ataque lo presentó Kortchinsky [43]. Dada la complejidad de este tipo de ataque, a veces lo que se logra es un *crash* del sistema operativo invitado, logrando así

una denegación de servicio. La figura a continuación muestra la relación entre los recursos virtuales y los físicos y cómo un atacante lograría sus objetivos.

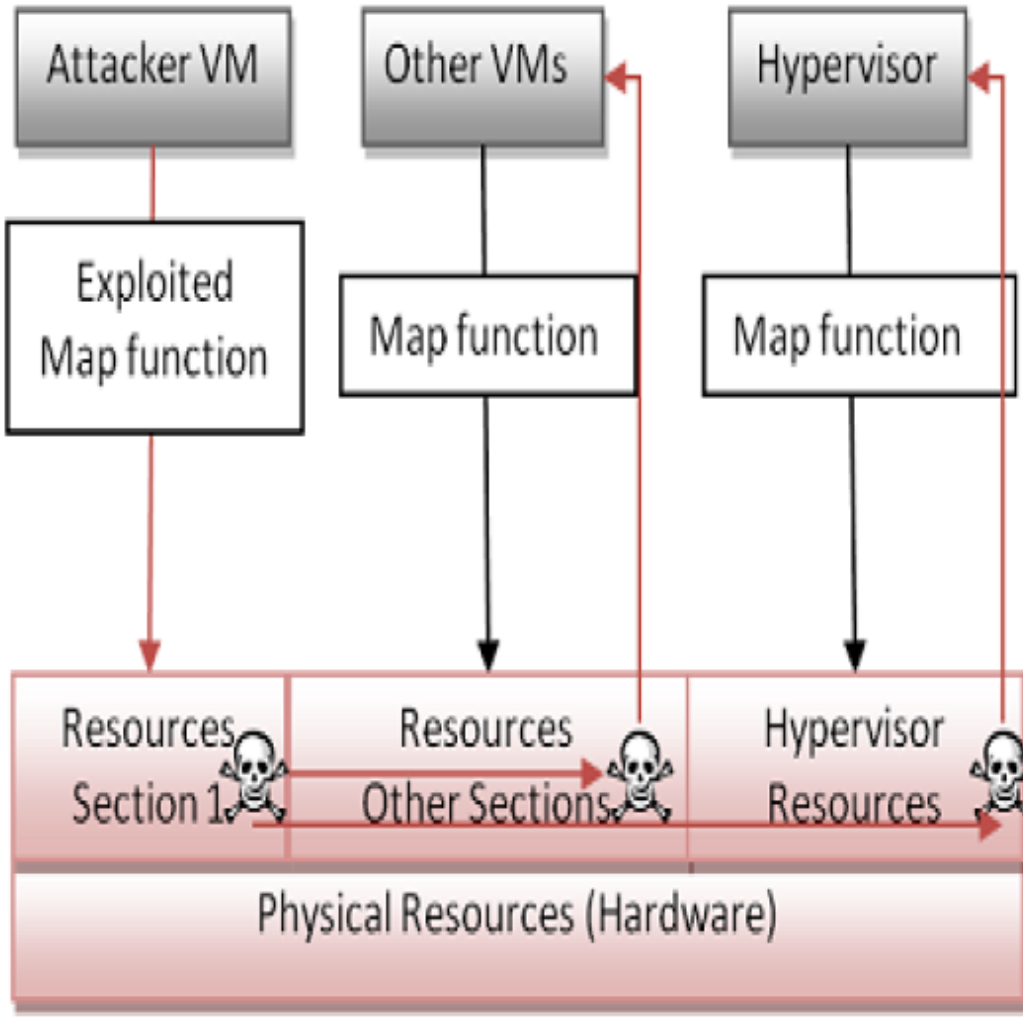

Figura 6. Ataque al hipervisor a partir del sistema operativo invitado. Fuente: [32]

6.3 *Check-out* de librerías virtuales

Este ataque se produce cuando se hace un *check-out*[29] de una imagen de MV, para luego infectarla en un hipervisor. Cuando posteriormente se vuelve a guardar en el repositorio, la imagen queda comprometida y se puede

[29] En este contexto, un *ckeck-out* se refiere a cuando se adquiere una copia de un recurso (la imagen en este caso) de un repositorio.

expandir a otros hipervisores [50]. Este ataque explota el hecho de que el hipervisor de MVs podría estar menos protegido que el repositorio original.

6.4 Ataque de migración

Este ataque se produce en la red durante la migración de una MV de un lugar a otro. Este ataque saca provecho de la facilidad con que se puede mover las MVs a través de la red [37], ya sea para hacer *backup*, distribuir actualizaciones, compartir imágenes, etc. El atacante puede ganar acceso no autorizado a las MVs, acceder a datos sensibles o instalar código malicioso.

Un caso particular de este tipo de ataque ocurre a partir de una funcionalidad que brindan algunos hipervisores. Se la llama *migración de una MV iniciada* (*live virtual machine migration*). Es una característica que permite migrar un sistema virtual de un hipervisor a otro (incluso entre hipervisores que estén en distintos equipos físicos) sin apagar el sistema. Esta funcionalidad facilita la administración y podría utilizarse, entre otras cosas, para balancear la carga de trabajo de un servidor. El mecanismo de comunicación empleado para transferir el estado de un hipervisor a otro debe ser autenticado (tanto en el origen como en el destino), confidencial y resistente a modificaciones (estas dos últimas características se consiguen mediante el uso de criptografía).

Un ejemplo de ataque a esta funcionalidad fue presentado en 2008 en *Black Hat* por Oberheide et al. [51]. Los investigadores desarrollaron una herramienta, llamada Xensploit, para modificar la memoria de las MVs en tránsito, que funciona con los productos de virtualización Xen (versión 3.1.0) y VMware (Virtual Infrastructure 3). Xensploit realiza un ataque *Man in the Middle*[30] donde manipula la memoria de la MV interceptada en la red durante la migración. En los ejemplos que aportan solo manipulan la memoria como

[30] En un ataque *Man in the Middle* (hombre en el medio) el atacante se sitúa en el medio de un canal de comunicación (de ahí el nombre del ataque) y puede leer, modificar o insertar tráfico. Este ataque a la confidencialidad e integridad de la comunicación se puede prevenir mediante el uso de canales de comunicación seguro (que utilizan cifrado, funciones de *hash*, certificados, etc. para asegurar el canal).

prueba de concepto, pero afirman que se podría utilizar el ataque para insertar un *rootkit* en la máquina migrada.

Los ataques de migración y de *check-out* de librerías virtuales están relacionadas con una de las características vistas de las MV, la movilidad. Esta característica, además, está vinculada al robo de MVs (en inglés, *Virtual machine theft*) [52]. Como una MV es un conjunto de archivos, la MV entera puede ser copiada o removida del hipervisor o del lugar donde se almacenan las MVs. Este riesgo se debe manejar con mecanismos de control de acceso y cifrado de las MVs.

6.5 Hipervisor como código malicioso y *rootkit*

El hipervisor puede ser usado por algún código malicioso o *rootkit* para instalarse a sí mismo como hipervisor por debajo de un sistema operativo (que es la víctima del ataque). La detección del código malicioso, en este caso, es muy difícil, ya que el *malware* puede interceptar todas las operaciones del sistema operativo sin que el antivirus observe nada sospechoso. Si bien los métodos tradicionales para la detección no funcionan, se pueden realizar análisis más complejos, desde el propio sistema infectado, para encontrar la anomalía. Estos métodos consisten en buscar alguna de las perturbaciones que produce el hipervisor, por ejemplo, el tiempo del procesador, la memoria, espacio en disco o el uso de dispositivos específicos.

Una de las primeras implementaciones de este concepto se atribuye a un trabajo hecho por investigadores de Microsoft y de la Universidad de Michigan, que desarrollaron el *rootkit*[31] SubVirt. Luego, la investigadora Joanna Rutkowska[32] desarrolló Blue Pill, que busca mejorar SubVirt. A continuación, veremos ambos productos.

---

[31] Un *rootkit* es un programa con acceso administrativo a un sistema, que se oculta a sí mismo alterando el funcionamiento normal del sistema operativo. Si bien puede utilizarse para distintos fines, generalmente es usado por atacantes que buscan acceder y controlar el equipo sin ser detectados.

[32] Joanna Rutkowska también es autora de Red Pill, una herramienta que al correrse permite detectar si se está ejecutando en una máquina real o virtual. Otra herramienta relacionada con seguridad y virtualización creada por esta investigadora (junto a Rafal Wojtczuk) es

6.5.1 SubVirt

SubVirt es un nuevo tipo de código malicioso, llamado por sus autores *rootkit basado en MVs* o *Virtual Machine Based Rootkit* (VMBR, por sus siglas en inglés) [53]. Fue desarrollado por investigadores de Microsoft y de la Universidad de Michigan y presentado en 2006. Esta herramienta instala un hipervisor y mueve el sistema operativo original (el que se quiere atacar) al entorno virtual controlado por el hipervisor. En la siguiente figura se ilustra la idea.

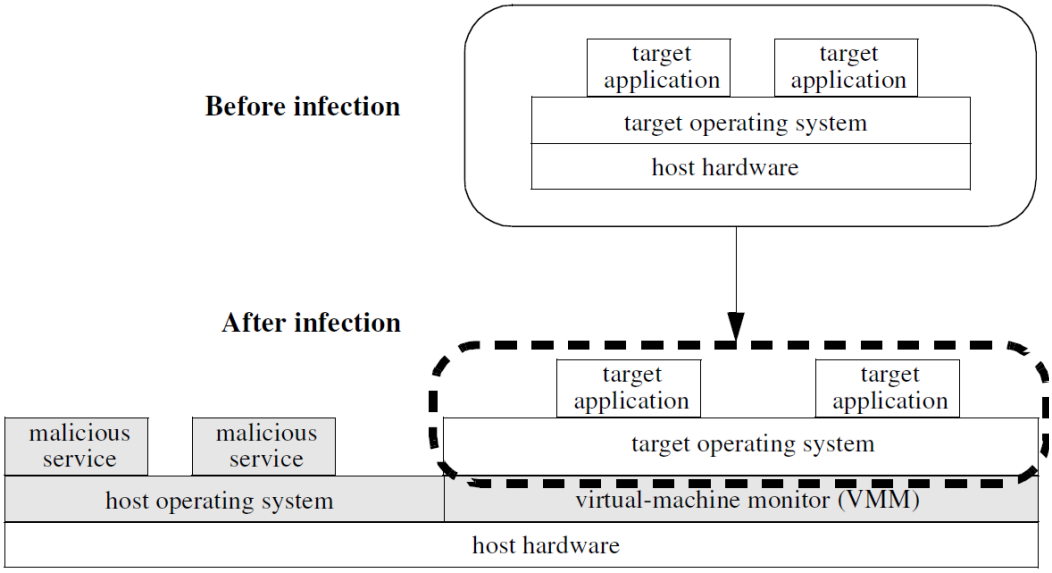

Figura 7. Muestra cómo queda un sistema luego del ataque de SubVirt. Fuente: [53]

Una vez que el programa mueve el sistema operativo objetivo sobre el hipervisor, este aísla completamente el estado del *malware* y sus eventos del sistema objetivo. Gracias a esto, los programas del sistema objetivo no pueden ver ni modificar el código malicioso. Mientras tanto, el hipervisor puede ver, e incluso modificar, todo el estado y los eventos del sistema objetivo (teclas apretadas, paquetes de red, archivos, etc.). Este tipo de ataque es muy difícil de detectar y de eliminar, ya que no puede ser detectado por programas que corran sobre el sistema operativo afectado.

En su trabajo, los investigadores aportaron una prueba de concepto en el sistema operativo Linux (con el hipervisor VMware) y otra en Windows (con el

---

Qubes, un sistema operativo centrado en la seguridad a partir del aislamiento basado en lo que llamaron *MVs descartables*. Lanzaron su primera versión Beta en 2010. La última versión estable, hasta el momento de escribir este trabajo, es la 4.0.1 publicada a comienzos de 2019.

hipervisor VirtualPC). Luego de mover los sistemas atacados sobre el hipervisor, corrieron servicios maliciosos (por ejemplo, un detector de teclas apretadas y un buscador de información sensible en los archivos) para mostrar el daño que pueden causar estos ataques.

Para instalar un VMBR en una computadora, el atacante debe tener acceso al sistema con los privilegios suficientes como para modificar la secuencia de arranque (*booteo*), ya que allí debe insertarlo. El VMBR debe quedar instalado en algún sistema de almacenamiento persistente (comúnmente el disco rígido). Una vez instalado e incorporado a la secuencia, al encender el equipo carga primero el VMBR. Luego el hipervisor carga el sistema operativo atacado, que se ejecuta de manera normal, sin enterarse que está sobre el VMBR. A continuación, los servicios maliciosos pueden ejecutarse de manera transparente para la víctima.

6.5.2 Blue Pill

El nombre Blue Pill hace referencia a un concepto de la película Matrix y la idea de la pastilla azul y la roja. La primera representa una “ignorancia feliz” y la segunda la “dolorosa verdad” de la realidad. En este caso, la pastilla roja es una técnica para detectar la presencia del hipervisor y la pastilla azul se usa para la infección de la máquina.

En la conferencia de *Black Hat* de 2006, la investigadora Joanna Rutkowska presentó un *rootkit* denominado Blue Pill [54]. En su primera versión, esta herramienta aprovechaba una funcionalidad nueva de los procesadores de AMD que sirve para optimizar la virtualización (AMD-V Pacifica), luego agregaron soporte para los procesadores de Intel. Esta herramienta lo que hace es mover el sistema operativo a atacar a una MV. Provee un hipervisor liviano para controlar el sistema movido. Una vez migrado, el hipervisor puede monitorear y modificar el sistema operativo que corre sobre él, como lo hace SubVirt.

El propósito de Blue Pill, según su autora, es desarrollar un programa malicioso que no pueda ser detectado, aunque el código del algoritmo sea conocido. Blue Pill no modifica el disco rígido ni se instala en el equipo, de hecho, si este se reinicia, Blue Pill desaparece ya que no es persistente. Como

no queda instalado, no puede ser detectado con el sistema apagado, algo que sí ocurre con SubVirt. Como los programas que podrían detectar esta situación corren sobre el hipervisor, el *malware* puede ocultar sus actividades [53]. Por ello, Joanna Rutkowska afirmaba que su sistema era 100% indetectable. Esta afirmación generó polémica entre algunos investigadores. Thomas Ptacek (de *Matasano* Security), Nate Lawson (de *Root Labs*) y Peter Ferrie (de Symantec) la desafiaron a probarlo, pero este desafío no se realizó. Al año siguiente, Rutkowska ya no afirmaba que Blue Pill era 100% indetectable, sino que era muy difícil detectar [55].

A diferencia de SubVirt, Blue Pill no necesita modificar la secuencia de arranque del equipo. Tampoco requiere reiniciar el sistema operativo atacado. Blue Pill no virtualiza los dispositivos, algo que sí hace SubVirt. Al no virtualizarlos evita la sobrecarga que ello genera y elimina alguno de los métodos usados para detectar la virtualización (estos métodos se verán más adelante).

6.6 Uso de canales encubiertos

Un canal encubierto es una vía para comunicar información entre dos partes a través de un canal oculto, no diseñado para tal fin. Lo que se busca al usar un canal encubierto es que la comunicación sea realizada en secreto, saltándose los mecanismos de control que pudieran existir. Para ello se usa una entidad no pensada para comunicar, se busca de este modo que nadie se entere que la comunicación está teniendo lugar. Los canales encubiertos son un tipo de fuga de información [56].

En un entorno con más de una MV estas no se deberían comunicar entre sí, salvo que en el hipervisor que las contiene se otorguen los permisos necesarios (por ejemplo, se puede configurar la red para que se vean entre las MVs, para que compartan una carpeta o el portapapeles). En estos casos permitidos, la comunicación se realiza mediante el hipervisor y bajo su control. Como una de las características fundamentales de las MVs es su aislamiento del entorno, si se produce la comunicación por un canal encubierto se rompe esta propiedad, lo que representa una amenaza.

La infraestructura de virtualización, donde se comparten los recursos de *hardware*, es propensa a los canales encubiertos [57]. A pesar de los esfuerzos para evitarlos, se han presentado numerosos trabajos donde se logra usar estos canales. A continuación, se presentan algunos ejemplos.

En 2006 Wang et al. [56] mostraron cómo realizar canales encubiertos entre MVs a partir de funcionalidades de la arquitectura del procesador (*multithreading*). Logra realizar la comunicación entre procesos que se ejecutan en paralelo en distintos hilos del procesador. Cuando el emisor quiere transmitir un '1', ejecuta una operación que hace uso de unidades funcionales del procesador. Si quiere transmitir un '0' ejecuta varias instrucciones NOP. El receptor puede detectar el valor transmitido midiendo el tiempo que le lleva a él realizar determinadas operaciones.

En 2009, Ristenpart et al. [10] presentaron un trabajo donde realizan un canal encubierto, a partir del tiempo de lectura del disco rígido. El emisor interactúa con el disco o no hace nada según quiera transmitir un '0' o un '1'. En simultaneo, el receptor calcula el tiempo que necesita para leer en el disco. Este tiempo depende de lo que el emisor está haciendo con el disco. Según cuán largo sea ese tiempo, el receptor detecta si se envió un '0' o un '1'. Este canal encubierto es usado en el mismo trabajo [10] para detectar corresidencia entre distintas máquinas virtuales que usan infraestructura en la nube. Además, crean otro canal encubierto, con mejor velocidad de transmisión, mediante el uso de la *cache*. El emisor interactúa con la *cache* para influir en el tiempo requerido por el receptor para leer en la *cache*. Según el tiempo que necesita para leer, el receptor distingue si el *bit* transmitido es un '0' o un '1'. En el Anexo B se muestra conceptualmente el algoritmo usado para lograr la comunicación. Estos canales encubiertos fueron probados en el servicio de la nube de Amazon EC2.

En 2010 Okamura et al. [58] presentaron una herramienta llamada CCCV (*Covert Channels using CPU loads between Virtual machines* o, en castellano, canal encubierto que usa la carga del CPU entre MVs). Como su nombre lo indica, esta herramienta crea un canal encubierto entre distintas MVs usando la carga del procesador. Una ventaja de este mecanismo es que para cambiar la carga de un CPU no se requiere un usuario con privilegios especiales. Para funcionar es necesario que las MVs estén en el mismo hipervisor (las pruebas

las realizaron con un hipervisor Xen versión 3.1.4) y que estén en el mismo *core*[33] del procesador. En el trabajo describen cómo se podría extender CCCV para que funcione incluso si las MVs están en distintos *cores*. Para comenzar la transmisión, primero se deben sincronizar el emisor y el receptor. Para ello, el emisor genera una carga del CPU que sigue un patrón predefinido. Cuando el receptor detecta el requerimiento de la comunicación, hace una pausa por un tiempo predefinido, también realizada por el emisor. Luego comienza la fase de transmisión, donde se envía *bit* a *bit*. La distinta carga del CPU se usa para identificar si el *bit* es un '0' o un '1'.

En 2011 Xu et al. [59] presentaron un nuevo canal encubierto, que usa el tiempo de acceso a la *cache* L2 para lograr la comunicación. Logran mejorar la tasa de transferencia obtenida por Ristenpart et al. [10], gracias a un procedimiento más eficiente usado para codificar los *bits*. A pesar de la mejora obtenida, analizan la baja tasa de transferencia que tienen los canales encubiertos conocidos. Concluyen que la tasa no permite grandes fugas de información, pero sí es suficiente como para transferir información sensible de pocos *bits*, por ejemplo, números de tarjetas de crédito o claves privadas.

En 2012 Wu et al. [60] presentan un nuevo canal encubierto basado en el tiempo de acceso a la *cache*. Con un nuevo esquema de codificación logran mejorar la ratio de transferencia de los trabajos previos. Además, presentan otra canal que usa el bus de memoria como medio para la transmisión. Básicamente, ponen el bus en estado *ocupado* o *libre* según quieran enviar un '0' o un '1'. La ratio de este canal es más baja que el que usa la *cache* presentado en el mismo trabajo, pero presenta como ventaja que no requiere que las MVs estén en el mismo procesador (sí requerido con el método que usa la *cache*). Ambos canales encubiertos fueron probados en el servicio de la nube de Amazon EC2. El ancho de banda obtenido en este escenario real les permite concluir que los canales encubiertos en la nube sí son una amenaza que debe tenerse en cuenta, debe analizarse y mitigarse.

[33] Un *core* (núcleo en castellano) es una unidad de procesamiento dentro del procesador. Los procesadores modernos tienen varios *cores* en su interior. Su objetivo es acelerar la velocidad gracias al procesamiento en paralelo.

En 2015 Maurice et al. [61] consiguen una nueva mejora en la ratio, a partir de su herramienta C5. Esta herramienta establece un canal entre distintas MVs que corren sobre distintos *cores* del mismo procesador. Para la comunicación usan el tiempo de demora de acceso a la *cache* de nivel 3[34]. A diferencia de los trabajos previamente presentados ([10, 58, 59, 60]), aquí no se requiere que las MVs corran sobre el mismo *core*, facilitando así su uso. En la siguiente figura se observan las características de la arquitectura utilizada en este ataque.

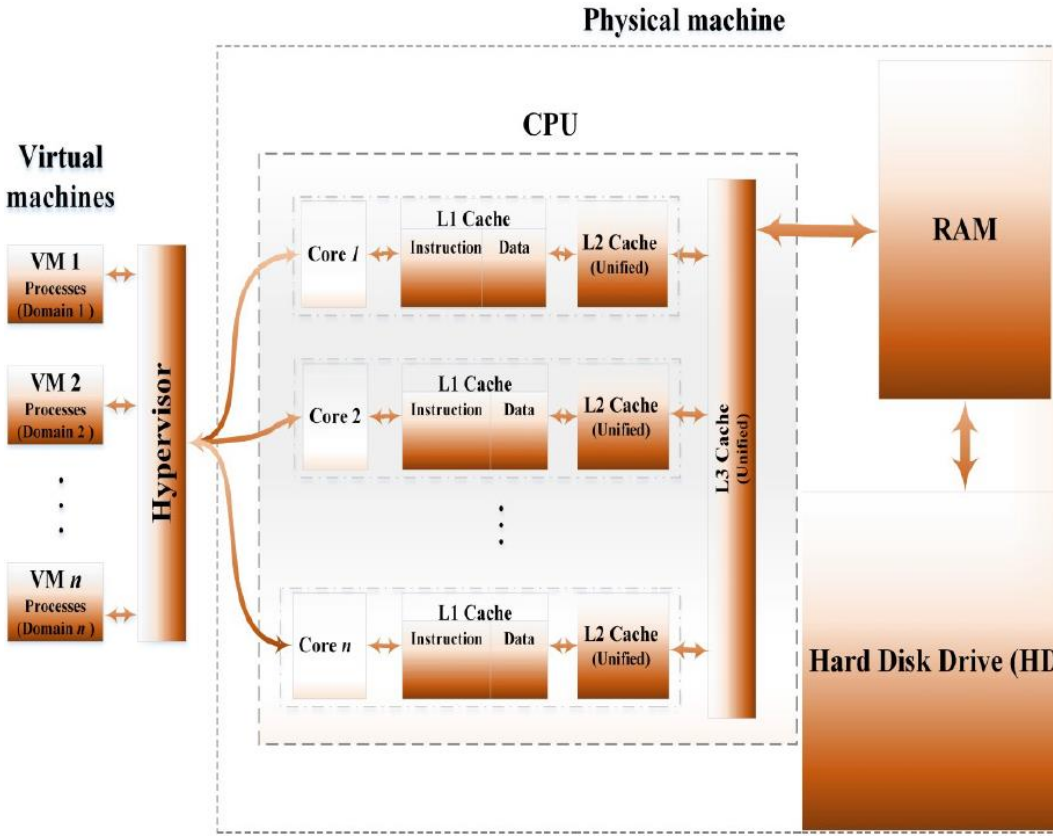

Figura 8. Distintas MVs que comparten la misma memoria L3. Fuente: [62]

[34] Las *cache* de nivel 1 y 2 son propias de cada *core*. En cambio, la de nivel 3 es compartida por los distintos *cores* del procesador y cada uno de ellos puede direccionar toda la *cache* [61]. De los tres niveles, la 3ª es la que tiene mayor tamaño, usualmente de varios megabytes. Por ejemplo, el procesador de Intel Xeon E5-2666, presentado en 2014, tiene 25 megabytes de *cache* nivel 3.

6.7 Ataque de canal lateral

En los ataques de canal lateral se usa información obtenida gracias a efectos laterales (de allí su nombre) de la implementación física de un sistema. La electricidad consumida, el ruido, el calor o las vibraciones generadas por un procesador son ejemplos de efectos laterales, que pueden generar una fuga de información.

Previamente se comentó la problemática que surge de la corresidencia[35], donde distintas MVs comparten un mismo componente de *hardware* entre ellas. Los problemas surgen debido a que los componentes (por ejemplo, la *cache* del CPU y la GPU) no fueron diseñados teniendo en cuenta el aislamiento requerido en virtualización. Una problemática similar ocurre al utilizar bibliotecas de *software* que no fueron diseñadas para su uso en ambientes compartidos. Estos componentes de *hardware* y *software* son usados como canal lateral para desde una MV obtener información de otra MV.

En 2009 Ristenpart et al. [10] presentaron un trabajo donde realizan un ataque de canal lateral, a partir de la carga de trabajo de la *cache* compartida entre la víctima y el atacante. Muestran cómo en la nube (en particular en el servicio de Amazon EC2) se generan nuevas vulnerabilidades debido a que se comparte la infraestructura física entre distintos clientes. Analizan cómo se puede obtener información sobre la infraestructura de la nube, detectar dónde reside una determinada MV (posible objetivo de un ataque) y generar nuevas instancias de MVs, controladas por el atacante, hasta crear una en el mismo lugar físico que la MV objetivo. Una vez lograda esta corresidencia, se realiza el ataque de canal lateral. La MV maliciosa puede detectar cuándo otra instancia de MV está realizado procesamiento. A partir de la medición del uso de la *cache* compartida, puede medir la utilización del CPU y estimar la carga de trabajo de la MV corresidente. Esta información la usa para verificar la corresidencia entre MVs, detectar aumento en el tráfico que recibe una web

[35] Hay ataques, en la nube, donde el atacante busca la corresidencia, mediante la creación de una MV bajo su control en el mismo lugar del objetivo a atacar. Una vez lograda la corresidencia se intenta el ataque. Ristenpart et al. [10] muestran cómo lograron la corresidencia en la nube de Amazon (EC2) y luego realizaron ataques de canal encubierto y de canal lateral.

corresidente y para calcular el tiempo entre pulsaciones del teclado en la MV víctima.

En 2011 Hlavacs et al. [63] presentaron otro ejemplo de ataque de canal lateral. Para lograrlo usaron, como canal lateral, el consumo de energía que se guardaba en un archivo de *log*. Otro ejemplo de ataque fue presentado en 2012 por Zhang et al. [9]. Allí lograron obtener la clave privada de una implementación[36] de ElGamal[37] desde una MV usada por el atacante a otra que hace el cifrado. El tipo de ataque utilizado se denomina *ataque de canal lateral entre MVs* (en inglés llamado *cross virtual machine side-channel attack*. Si el ataque es entre MVs, como en este caso, el atacante busca alguna información generada por la víctima de otra MV que comparte el *hardware*. El ataque presentado por Zhang et al. [9] se basó en el tiempo de acceso a la memoria *cache* usado por el algoritmo. Este tipo de ataque de canal lateral se llama *time-driven* o *time-driven cache*.

Irazoqui et al. [64] presentaron otro ataque en 2014. Allí lograron obtener la clave de una implementación de AES[38], a partir del tiempo de acceso a la memoria *cache* usado por el algoritmo[39]. Como la *cache* es uno de los dispositivos más compartido entre los procesos, suele ser un objetivo para ataques de canal lateral [65].

Un extenso análisis sobre estos tipos de ataque, con detalles de bajo nivel, es presentado por Rebeiro et al. [66]. Anwar et al. [65] publicaron una amplia revisión de la literatura (presentada hasta 2017) donde analizan los ataques de canal lateral. Allí también se los clasifica en varios tipos, se estudia estos ataques en MVs, se analizan mecanismos de prevención y se dan recomendaciones para prevenir estos ataques.

---

[36] La biblioteca utilizada en este ataque es libgcrypt v.1.5.0, incluida en Gnu Privacy Guard (GnuPG) v.2.0.19

[37] ElGamal es un muy usado algoritmo criptográfico para firmar o cifrar mensajes. Fue creado en 1984 por Taher Elgamal.

[38] AES son las siglas en inglés de *Advanced Encryption Standard*. Es un algoritmo de cifrado simétrico ampliamente utilizado desde que fue anunciado como estándar por el gobierno de Estados Unidos en 2001.

[39] El algoritmo utilizado es el implementado en OpenSSL 1.0.1f. En el ejemplo se usó VMware ESXI 5.5.0 build number 1623387 y un Ubuntu 12.04 64-bits como sistema operativo invitado.

6.8 Ataque de *rollback* de máquina virtual

Otro tipo de ataque es presentado por Khalil et al. [6]. En su trabajo hacen uso de la funcionalidad que tienen los hipervisores para generar *snapshots*, con el estado actual del procesador, los discos y la memoria. Si bien esta funcionalidad es muy útil para mantenimiento de MVs y para dar soporte a sistemas tolerante a fallas, presenta una oportunidad de ataque llamada *rollback* de MV.

Un atacante puede reemplazar un sistema con un *snapshot* viejo, para eliminar parte o todo el registro de lo sucedido entre el momento en que se creó el *snapshot* y el momento actual del sistema. Un ejemplo donde es útil este tipo de ataque ocurre en un escenario donde un atacante quiere, por fuerza bruta[40], adivinar una contraseña de un usuario. Si el sistema donde se quiere autenticar se bloquea luego de una determinada cantidad de intentos, el método de fuerza bruta no sirve, pero si realiza este tipo de ataque, se puede volver al estado anterior de la MV luego de cada bloqueo. De esta manera reinicia el contador de intentos y se saltea así este mecanismo para evitar los ataques de fuerza bruta.

6.9 Detección de entornos virtuales

Un sistema operativo invitado corre en un entorno emulado y proporcionado por el hipervisor, con acceso tanto virtual como real al *hardware*. En teoría, el entorno provisto es autocontenido, aislado e indistinguible de una máquina real. Los investigadores de código malicioso suelen hacer uso de MVs para analizar dichos códigos. Además, las MVs suelen usarse para crear *honeypots*[41] y *honeynets*[42] que permiten hacer

[40] Se denomina *ataque de fuerza bruta* a un ataque en el cual se prueban todas las combinaciones posibles de caracteres hasta encontrar el valor buscado.

[41] Un *honeypot* es un sistema usado de señuelo para que sea víctima de un ataque. Simula ser un equipo vulnerable para permitir así ser atacado. El objetivo de esta herramienta usada en seguridad informática es detectar y analizar el comportamiento de los atacantes.

[42] Un *honeynet* es un tipo de *honeypot* especial que simula ser una red entera, pero está formada por *honeypots*. El objetivo suele ser el mismo que el de un *honeypot*, recopilar información sobre ataques.

análisis sobre los ataques. Por esto, para un atacante es muy importante detectar si su sistema objetivo está corriendo sobre una máquina real o virtual. Los desarrolladores de *malware* podrían tratar de frustrar el análisis de su código mediante la detección de entornos de virtualización. Si el *malware* detecta el entorno puede apagar cierta funcionalidad maliciosa de manera tal que a los investigadores se les dificulta su observación y análisis. Debido a su importancia, desde hace tiempo existen varias técnicas para verificar en qué tipo de entorno se encuentra corriendo una aplicación, tanto en forma local como de manera remota a través de una red.

Algunos de los métodos para la detección en forma local de MVs son buscar características de entornos virtuales en procesos, *file systems*, en el registro y en la memoria, buscar *hardware* virtual específico y buscar instrucciones de procesador específicas en entornos virtuales.

En 2004 Joanna Rutkowska presentó el código de su herramienta *Red Pill* que detecta si está corriendo en un entorno virtual [67]. La detección la hace mediante el uso de una instrucción del procesador (SIDT, *Store Interrupt Descriptor Table Register*), que devuelve valores constantes y conocidos para los virtualizadores.

Un caso de un *malware* que realiza este análisis es el gusano *Conficker*[43]. Este programa utiliza una variante de Red Pill para detectar dónde se está ejecutando. Si detecta que está en un ambiente virtual se apaga sin realizar ninguna otra acción, para dificultar el análisis del comportamiento del gusano.

También existen métodos y herramientas para detectar de manera remota la presencia de un hipervisor. Por ejemplo, Noorafiza et al. [30] hace la detección desde Internet a partir de análisis de patrones de tiempo de respuesta de paquetes IP. Franklin et al. [68] realizan el análisis mediante *Fuzzy benchmarking*, que trabaja haciendo un estudio del tiempo de la ejecución de un código en el equipo remoto (a partir de heurísticas usadas para reconocer el *hardware* del sistema remoto y del hipervisor). Jämthagen [69] trabaja analizando el comportamiento de las implementaciones NAT en los productos de virtualización.

[43] *Conficker* es un gusano que apareció en octubre 2008 y tuvo una difusión muy importante a principios de 2009. Explota una vulnerabilidad de desbordamiento de *buffer* de los sistemas operativos Windows (2000, XP, Vista, Server 2003 y Server 2008).

7 Recomendaciones

A continuación, se presentan las recomendaciones de dos organismos internacionales, el NIST y el ISO/IEC, sobre aspectos de seguridad que se deben tener en cuanta al utilizar la virtualización.

7.1 Del NIST

El NIST, por las siglas en inglés de *National Institute of Standards and Technology*, es una agencia que depende del gobierno de Estados Unidos. Tiene como objetivo promover la innovación en su país y la competencia de su industria. Para ello provee normas, guías y material de referencia de patrones en áreas como tecnologías de la información, biotecnología y nanotecnología. El *Information Technology Laboratory* es un laboratorio del NIST, que, entre otras áreas, se especializa en ciberseguridad. Ha realizado numerosos aportes, entre ellos las recomendaciones que se presentan en este capítulo [7].

A continuación, se describen tres particularidades que se deben tener en cuenta al usar virtualización.

Aislamiento del sistema operativo invitado. El hipervisor particiona y administra los recursos de *hardware* (disco, memoria, CPU, etc.) asignándoles una parte a cada sistema operativo invitado para que estos puedan usar sus propios recursos, mientras evita intromisiones de unos a otros. Esta separación ayuda a impedir accesos no autorizados y a prevenir inyecciones de código malicioso, tanto en los archivos como en la memoria de los otros sistemas. Además, estas particiones reducen la amenaza de denegación de servicio por el uso excesivo de recursos de los sistemas que comparten un mismo hipervisor. El aislamiento también incluye las limitaciones para que un sistema se comunique con otros sistemas virtuales, con el hipervisor y con el sistema operativo *host* (si está presente, como ocurre si el hipervisor es de tipo 2). Este aislamiento también ayuda a mitigar los *ataques de canal lateral* entre MVs, que pueden ocurrir cuando los equipos virtuales comparten el *hardware* subyacente. Si bien son difíciles de realizar (suelen requerir acceso físico directo al servidor) [7], podrían, mediante el análisis de patrones de uso

de algunos recursos (por ejemplo, de la memoria *cache*), revelar información sensible.

Muchos productos de virtualización proveen herramientas para compartir archivos, directorios y otros recursos entre los sistemas. Estos mecanismos de comunicación pueden ser un vector de ataque, por ello hay que configurarlos y usarlos correctamente, intentando minimizar su uso. Los sistemas de virtualización de tipo *bare metal* no incluyen estas funcionalidades.

Monitoreo del sistema operativo invitado. El hipervisor conoce completamente el estado de los sistemas que corren sobre él y, por lo tanto, tiene la capacidad para monitorearlos, lo que es conocido como introspección. Gracias a ello, puede hacer una auditoría sobre todo lo que pasa en el sistema invitado, tanto en la memoria, como el tráfico de red, uso del procesador, acceso a archivos y todos los componentes del sistema operativo. Muchos productos de virtualización proveen herramientas con controles de seguridad y acceso a la información obtenida con introspección. Por ejemplo, la información del *firewall*, del sistema de detección de intrusiones y control de acceso.

Administración de imágenes y *snapshot*. La gestión de estos *archivos contenedores*[44] (que incluyen el sistema operativo, las aplicaciones, los archivos de las aplicaciones y usuarios, su estado, la memoria) se debe realizar con especial cuidado, debido a la sensibilidad de su contenido. La problemática aquí es la misma que la señalada por Ritter [47] con respecto a la movilidad, es muy fácil replicar y mover estas imágenes, con lo cual se pueden generar copias de las mismas que no estén debidamente controladas. Para evitar problemas de integridad y confidencialidad se debe establecer cuidadosamente quién tiene permiso de acceso, modificación y reemplazo.

Más allá de los cuidados que hay que tener para proteger las imágenes, las ventajas de su uso pueden ser muy importantes. Por ejemplo, las imágenes sirven como *backup.* Si un sistema se compromete, de manera rápida y sencilla se puede reemplazar por la última imagen que funcionaba

[44] Los *archivos contenedores* son archivos que internamente contienen otros archivos y directorios. Ejemplos de este tipo de archivos son los *zip*, *jar*, *tar*, *iso*, etc.

correctamente. La imagen comprometida puede preservarse para luego ser sometida a un análisis forense que permita detectar el origen del problema y el alcance del mismo.

Otro ejemplo de las ventajas del uso de imágenes es cuando se hace la instalación de un sistema operativo junto con varias aplicaciones, se configura y se prueba en una sola MV (*golden image*). Luego de su verificación, esta imagen se distribuye a un conjunto de *host* y usuarios. Esto permite ahorrar tiempo y costos y aumenta la consistencia entre los distintos *hosts* que usan esta imagen.

Dentro de una organización, con el crecimiento de la virtualización tanto para *servers* como para *desktop*, la gestión de las imágenes genera un desafío importante. Por ello, algunos productos de virtualización proveen soluciones para su administración, que pueden examinar las imágenes guardadas y actualizarlas cuando sea necesario, tanto para aplicar parches con actualizaciones como para cambios en la configuración de la seguridad. Las imágenes pueden ser monitoreadas para detectar cambios en los archivos, usando funciones de *hash*[45] criptográficas. Para ello se debe almacenar, en algún lugar seguro, el valor de *hash* de cada imagen validada. Cada vez que se transfiere o se va a utilizar una imagen se debe recalcular su valor de *hash* y compararlo con el valor de *hash* previamente validado. Si encuentra cambios se debe verificar el origen del mismo y determinar si es correcto o no. El análisis de las imágenes puede servir para detectar *rootkits* y otro código malicioso.

La recomendación es minimizar la creación de imágenes y *snapshot* e implementar un proceso para su gestión, que establezca una política que gobierne su creación, su seguridad, su distribución, su uso, su almacenamiento, su retiro y su destrucción.

[45] Una función *hash* es un algoritmo que, a partir de un bloque de datos, genera como salida una cadena de caracteres de longitud fija. Esta salida se considera un resumen o una huella del archivo de entrada. Para que el algoritmo de *hash* se considere criptográfico debe cumplir algunas propiedades. Se debe garantizar que a partir de un valor de *hash* no se pueda calcular el valor del archivo que lo genera. Otra propiedad fundamental que debe cumplir el algoritmo es que debe ser computacionalmente imposible encontrar dos archivos distintos que generen el mismo valor de *hash*. Un uso que se le da a la función de *hash* es para comprobar si un archivo fue modificado, comparando no los archivos en sí mismos sino sus valores de *hash*.

Los usos que se le puede dar a la virtualización varían entre las distintas organizaciones. Por ello, los controles de seguridad pueden cambiar de acuerdo al uso y al proveedor de la solución elegida. A continuación, se describen recomendaciones comunes en virtualización para reducir los riesgos.

Proteger todos los componentes que forman parte de la solución de virtualización utilizada. Todos los componentes deben ser tenidos en cuenta: el hipervisor, los sistemas operativos usados (tanto los sistemas operativos invitados como los de *host*, si existen), las aplicaciones usadas y el sistema de almacenamiento. Para asegurar estos componentes se deben utilizar las buenas prácticas de seguridad que se utilizan en ambientes no virtualizados: aplicaciones actualizadas con los parches, configuraciones seguras, utilización de *firewalls*[46], antivirus y otros mecanismos para detectar y parar ataques. Estas prácticas se deben aplicar tanto en el ambiente virtual como en el real donde corre el hipervisor.

Restringir y asegurar el acceso del administrador a la solución de virtualización. Controlar el hipervisor implica controlar las MVs que ahí corren. De ahí la importancia de restringir al máximo las autorizaciones para acceder al mismo. En caso de que se acceda remotamente al hipervisor, también se debe proteger la comunicación al mismo mediante un canal seguro.

Asegurar el hipervisor. Para asegurar el hipervisor se deben realizar algunas tareas similares a las que se realizan en cualquier otro *software*, por ejemplo, aplicar los parches con las actualizaciones. Además, hay tareas propias del hipervisor como deshabilitar *hardware* virtual y servicios que no se usen. Se pueden usar las propias herramientas del hipervisor para monitorear las máquinas virtuales allí alojadas. Se deben tener en cuenta medidas de seguridad física para controlar el acceso al equipo donde corre el hipervisor.

Antes de la instalación, planificar minuciosamente toda la solución de virtualización. La seguridad debe ser tenida en cuenta desde el inicio del proyecto de virtualización para maximizar la seguridad y minimizar los costos.

---

[46] Un *firewall*, también llamado *cortafuegos* en castellano, es una herramienta utilizada para controlar el tráfico de una red. Permite bloquear el tráfico en caso de que detecte que no fuera permitido. Se puede implementar tanto en *software* como en *hardware*.

En la planificación se deben tener en cuenta todas las políticas relevantes de la organización y se debe cumplir con ellas.

7.1.1 Hacer más segura la virtualización

En principio, para aumentar la seguridad de la virtualización, primero hay que seguir los mismos pasos que se aplican cuando no se usa virtualización. Esto es, definir una política de seguridad con buenas prácticas, restringir el acceso a las herramientas administrativas y a los archivos (siguiendo el principio de *menor privilegio*[47]), mantener el *software* actualizado con los parches de seguridad, usar *baseline* de configuración seguros, hacer monitoreo, auditoría y análisis de *log* de todas las capas de la solución, usar *firewalls*, antivirus y capacitar a los usuarios. Se debe seguir una estrategia de defensa en profundidad[48] [46]. Si bien estas recomendaciones no son suficientes, sí son el primer paso.

El programa que controla el hipervisor, debería asegurarse con los mismos mecanismos que se asegura cualquier programa (control de acceso lógico, control de seguridad física sobre el equipo donde está instalado el hipervisor, monitoreo de los *logs* que genera, etc.), teniendo en cuenta que toda la infraestructura virtual depende de él. El acceso a la administración de la virtualización se debe otorgar solo a usuarios administradores. Si se habilita el acceso remoto, debería configurarse correctamente un *firewall* que restrinja el acceso y permita acceder solo a usuarios autorizados.

A continuación, se presenta un resumen de las recomendaciones del NIST [7] sobre seguridad para el hipervisor:

---

[47] El principio de menor privilegio busca que tanto usuarios como procesos tengan acceso solo a los recursos necesarios para realizar su tarea, por el tiempo requerido y solo con el derecho necesario (de lectura, escritura, etc.).

[48] El concepto de defensa en profundidad tiene un origen militar y se refiere a colocar varias líneas defensivas sucesivas en vez de una única línea muy fuerte. En seguridad informática es un modelo que busca proteger los recursos en diferentes capas. El objetivo es que si un atacante logra vulnerar una capa aún quedan otras que protejan el activo. Un ejemplo de protección de un archivo es protegerlo con permisos o criptografía, asegurar la aplicación que accede al mismo, el equipo, la red interna, el perímetro lógico y el físico donde está almacenado.

- Instalar todas las actualizaciones del fabricante del producto de virtualización. La mayoría de los hipervisores incluyen una funcionalidad que busca e instala las actualizaciones disponibles de manera automática.
- Restringir el acceso a las interfaces de administración. Proteger todos los canales de comunicación, usando módulos criptográficos seguros.
- Sincronizar los relojes de toda la infraestructura de virtualización con un reloj común.
- Desconectar todo el *hardware* físico que no se use, por ejemplo placas de red, disqueteras[49] o medios de almacenamiento removibles.
- Deshabilitar todos los servicios que no sean estrictamente necesarios, como, por ejemplo, el portapapeles o archivos compartidos. Estos son un posible vector de ataque que conviene minimizar.
- Considerar el uso de la introspección para monitorear la seguridad de los sistemas operativos invitados y la actividad y comunicación entre MVs dentro del mismo hipervisor.
- Monitorear el hipervisor en sí mismo buscando signos de algún ataque. Esto incluye el control de la integridad de los archivos importantes y el análisis de *logs*.
- Proveer control de acceso físico al *hardware* donde corren los sistemas de virtualización y donde se almacenan los *backups* de imágenes.

En el caso de que el hipervisor corra sobre un sistema operativo *host*, todas estas consideraciones, también deben tenerse en cuenta con respecto a ese sistema operativo, ya que, si este se vulnera, se podría comprometer también al hipervisor.

En los sistemas operativos virtuales se deben seguir las siguientes recomendaciones.

[49] En 2015 se publicó una vulnerabilidad, llamada VENOM (*Virtualized Environment Neglected Operations Manipulation*), generada a partir de una falla en un controlador de disquetera virtual de QEMU. Esta falla existía desde 2004 en varias soluciones de virtualización (QUEMU, Xen, VirtualBox y KVM) y se solucionó luego de su publicación registrada como CVE-2015-3456.

- Administración de *log*. Se deben almacenar y analizar según la política definida.
- Se deben restringir los accesos remotos, salvo en casos estrictamente necesarios.
- Se debe definir y aplicar una política de autenticación.
- Mantener el sistema operativo actualizado. Lo mismo con las aplicaciones que corren sobre él.
- Definir y cumplir una política de *backups* para los sistemas.
- Deshabilitar *hardware* virtual no usado.
- Deshabilitar todos los servicios no utilizados.

7.1.2 Planificación e implementación de la seguridad virtual

Una correcta y segura implementación de una solución de virtualización requiere una atenta planificación. Se deben tener en cuenta las políticas de seguridad de la organización desde la fase inicial, para que los aspectos de seguridad sean tenidos en cuenta en todo el ciclo de vida del sistema.

Para lograr una implementación segura, NIST [7] considera 5 fases en el ciclo de vida del sistema, donde agrupa las tareas por etapas. Las sucesivas fases son 1) inicialización, 2) planificación y diseño, 3) implementación, 4) operación y mantenimiento y 5) finalización.

En la primera fase, inicialización, se debe identificar qué se busca lograr con la virtualización, analizar ventajas y desventajas de las distintas plataformas existentes, establecer una estrategia de alto nivel para las siguientes fases, definir qué aplicaciones se migrarán a la plataforma virtual y especificar los requerimientos funcionales y de negocio que deberá tener la solución a implementar. En esta fase se debe definir una política de seguridad para virtualización.

En la segunda fase, planificación y diseño, hay que elegir la solución de virtualización más adecuada según las necesidades relevadas en la fase anterior. Se debe especificar las características técnicas de la solución seleccionada, métodos de autenticación y los mecanismos criptográficos que se usarán tanto para la comunicación como para el almacenamiento de

archivos. Uno de los objetivos de esta etapa es la obtención de los componentes de la solución, tanto de *hardware* como de *software*.

En la tercera fase, la de implementación, se debe instalar, configurar y probar el sistema y las aplicaciones migradas. Luego se debe conectar con la red de producción. Se debe verificar que la solución, una vez instalada, cumpla con todos los requerimientos identificados y con la política de seguridad definida en la primera fase.

En la cuarte fase, la de operación y mantenimiento, se incluyen las tareas relacionadas con la seguridad que la organización debe realizar, incluyendo análisis de *logs*, detección de ataques, respuesta a incidentes, instalación de parches, etc. Hay que verificar que se estén aplicando todos los controles definidos.

En la última fase, la quinta, llamada de finalización, se incluyen las tareas a realizar cuando la solución de virtualización deja de utilizarse. Se debe analizar y determinar cuál será la información que se deberá preservar (por su posible utilidad o por cuestiones legales) y la eliminación definitiva de todos los demás datos.

Si bien el documento del NIST, comentado en este capítulo, tiene algunos años de antigüedad (es de 2011), sus recomendaciones no han perdido actualidad. En la norma ISO 21878, de fines 2018, que trata temas de seguridad en virtualización, se observa la vigencia del trabajo del NIST. En el capítulo siguiente, se analizará dicha norma ISO.

7.2 De la ISO/IEC

La ISO (por las siglas en inglés de *International Organization for Standardization*, Organización Internacional de Normalización) y la IEC (por las siglas en inglés de *International Electrotechnical Commission*, Comisión Electrotécnica Internacional) son organizaciones especializadas en la creación de estándares internacionales. En noviembre de 2019 se publicará una norma sobre virtualización, la ISO/IEC 21878:2018. Esta norma presentará pautas de seguridad para el diseño e implementación de servidores virtuales. A fines de 2018 se publicó un borrador final de esta norma

[29] que será el que se desarrollará en este capítulo. Las recomendaciones de esta norma son, en general, similares a las de NIST. A continuación, se repasa la norma ISO y se mencionan sus aportes.

La norma presenta una introducción a la virtualización, los tipos de virtualización (*full virtualization* y *paravirtualization*) y los tipos de hipervisores (tipo 1 y 2), comentados previamente en este trabajo. Luego, asume tres consideraciones del entorno operativo que no desarrolla, pero sí menciona y se deben tener en cuenta ya que son críticas. La primera es la seguridad física de los equipos donde están instalados los componentes usados en la virtualización. La segunda es la integridad de la plataforma, que no debe haber sido comprometida antes de la instalación de los componentes. La tercera menciona la fiabilidad de los administradores, que deben seguir las guías y políticas definidas en la organización.

Seguidamente trata las amenazas. Hace una distinción entre las amenazas comunes de cualquier infraestructura informática (virtualizada o no) y de las que surgen con la virtualización. Entre las primeras menciona los **errores de los administradores**, que involuntariamente pueden comprometer la seguridad. La **configuración insegura de la red**. El **compromiso de la plataforma** (por instalación de programas maliciosos o no confiables en el mismo entorno donde está el hipervisor). La **mala gestión de claves criptográficas**. Las **vulnerabilidades de programas de terceros** (independientes de los del hipervisor, como por ejemplo controladores de dispositivos). Los **accesos no autorizados** al sistema. Por último, el uso de **criptografía débil.**

Los riesgos específicos de la virtualización coinciden con los presentados por Cloud Security Alliance [49]. Los separan entre los de las MVs, los del hipervisor y los operacionales[50]. Entre los primeros están la **proliferación de MVs** (la ya comentada *VM sprawl*). Estás, como pueden ser no detectables, no se controlan ni se actualizan debidamente. La **información sensible en MVs**. La movilidad propia de estas máquinas aumenta el riesgo de compromiso de esta información. La **seguridad de MVs fuera de línea o**

[50] La norma agrega un cuarto conjunto de riesgos asociados a los servicios en la nube. No se incluyen en este trabajo ya que la nube está fuera del alcance de este trabajo.

inactivas. Estos equipos podrían quedar desactualizados en cuanto a parches y políticas de seguridad, quedando expuestos a vulnerabilidades cuando se vuelven a usar. La **seguridad en las imágenes pre configuradas**. Muchas organizaciones tienen imágenes maestras a partir de las cuales crean muchos equipos (*golden images*). Si estas imágenes son comprometidas los equipos creados a partir de ellas serán vulnerables. El último es el **agotamiento de recursos**. Se debe verificar que los procesos virtuales no hagan un consumo de recursos excesivo que pueda comprometer la disponibilidad de servicios. Por ejemplo, puede ocurrir este uso excesivo si en todas las MVs se corre un proceso a la misma hora, como puede ser un análisis del disco completo por el antivirus[51]

Define dos **riesgos del hipervisor**, uno es una **mala configuración**. Los hipervisores, en general, incluyen interfaces para administración. Estas incrementan la superficie de ataque y deben ser correctamente configuradas. El segundo riesgo son los **accesos no autorizados**. Recomienda no usar esquemas locales de autenticación, sino los centralizados en servidores externos.

Por último, agrega dos **riesgos operacionales**. El primero es el **secuestro de cuenta a través del portal de autoservicio**. Estos portales, si bien facilitan la gestión de los usuarios, puede generar riesgos que deben ser tenidos en cuenta. El segundo es la **carga de trabajo de distintos niveles de confianza en un mismo servidor**. Recomienda separar los sistemas según su nivel de confianza y agrupar en un mismo *host* físico sistemas del mismo nivel de confianza, evitando poner los de distinto nivel en el mismo *host*.

Luego de tratar el tema de riesgos, se presenta una serie de recomendaciones para asegurar todo el ciclo de vida de la virtualización. El ciclo de vida lo separa en cuatro fases (las mismas que las recomendadas por Cloud Security Alliance [49] y con las mismas tareas en cada una de las etapas). Las fases son preparación inicial, planificación y diseño,

[51] Para evitar este problema de los antivirus existen soluciones que mueven la protección por fuera de la MV a controlar, evitando así que haya un agente en cada MV. Estas herramientas liberan recursos del sistema, eliminan la posibilidad de que se los ataque desde dentro de la MV y resuelven el problema de agotamiento de recursos que podría generar un antivirus en cada MV [70]. Un ejemplo de esta solución es vShield Enpoint de VMware.

implementación y disposición final. Las tareas son similares a las presentadas por NIST.

Para la fase de planificación y diseño da una lista de requerimientos que se deben verificar:

1. ¿Está la MV aislada?
2. ¿Se controla la integridad del hipervisor?
3. ¿Se controla la integridad de la plataforma?
4. ¿Se hace control de acceso a las funciones de administración según lo planificado?
5. ¿Existen interfaces que facilitan las funciones de gestión?
6. ¿Se auditan los *logs* de las MVs y del hipervisor?
7. ¿El entorno de virtualización se configuró de acuerdo a las políticas de seguridad de la organización?
8. ¿Existe seguridad física a los equipos que dan soporte a la virtualización?
9. ¿Existen procesos de reclutamiento y capacitación que garanticen la confiabilidad de los administradores?

Para la fase de implementación da la siguiente lista de verificación:

1. ¿Se implementaron las políticas y controles para evitar la proliferación sin control de MV?
2. ¿La información sensible contenida en las MVs está protegida?
3. ¿Se implementaron las medidas de seguridad para las MVs fuera de línea e inactivas?
4. ¿Se implementaron las medidas de seguridad sobre las imágenes activas y las pre configuradas?
5. ¿Está asegurado el control y la visibilidad sobre el tráfico de las redes virtuales?
6. ¿Se implementaron las políticas y controles para prevenir el agotamiento de recursos?
7. ¿Se tomaron las medidas para asegurar al hipervisor?
8. ¿Se implementó un estricto control de acceso al hipervisor para prevenir accesos no autorizados?

9. ¿Se aplican los controles y políticas para prevenir secuestro de cuentas?
10. ¿Se usan las medidas de seguridad para separar por *host* físico los servicios de distintos niveles de confianza?
11. ¿Se aplican los procesos automáticos para verificar la publicación de parches para el hipervisor y los módulos de las MVs?

Por último, en la norma se presenta una detallada guía para poner en marcha la lista de verificación de la fase de implementación.

7.3 Comparación NIST - ISO/IEC

Los trabajos del NIST y de ISO/IEC son guías y recomendaciones que deben tenerse en cuenta al utilizar virtualización. Ambos trabajos contienen buenas prácticas, muy útiles para tener una lista de puntos básicos para trabajar con esta tecnología.

Los temas tratados por ambos organismos son similares, básicamente cambia la forma en que se agrupan las problemáticas y recomendaciones. Por ejemplo, mientras ISO/IEC separa el ciclo de vida en cuatro fases (inicialización, planificación y diseño, implementación y finalización), NIST, además de estas cuatro fases, agrega la de operación y mantenimiento (antes de la de finalización). Más allá de esta diferencia, las tareas en cada una de las fases son similares en ambas propuestas, con la diferencia de que ISO/IEC agrupa las tareas que NIST pone en implementación y operación en su fase de implementación.

En el trabajo de NIST se hace una introducción más amplia sobre la virtualización, comparado con la realizada por ISO/IEC. Otra diferencia es que ISO/IEC incluye riesgos específicos de los servicios en la nube que no son analizados en el trabajo del NIST[52]. Por último, ISO/IEC incluye una guía para

[52] NIST describe y analiza la problemática de la nube en otros trabajos (ver http://csrc.nist.gov/groups/SNS/cloud-computing/).

hacer una evaluación de riesgos para las MVs y una práctica y útil lista de verificación con controles para los riesgos identificados.

8 Resumen

Los ataques presentados en este trabajo son el fruto de la investigación de especialistas de informática y de seguridad, publicados en distintos medios (libros, revistas, web, conferencias, etc.). El objetivo de estas investigaciones (por lo menos las aquí presentadas) no es producir ataques a empresas u organizaciones, sino advertir a la comunidad de las debilidades que pueden surgir al utilizar la virtualización y distintos aspectos que se deben considerar para evitar incidentes de seguridad. Estos trabajos suelen incluir remediaciones y, dependiendo del tipo de debilidad presentada, pueden generar parches de los proveedores de productos de virtualización. Además, suelen influir en cambios de diseño en futuras implementaciones y en la forma de operar en estos ambientes, para evitar las debilidades publicadas. Concretamente, gracias a todos esos trabajos e investigaciones (y a muchos más no comentados aquí), las herramientas de virtualización han corregido gran parte de los problemas detectados y mejorado notablemente su seguridad. Asimismo, han colaborado a mejorar los procedimientos para operarlas de manera segura.

Al utilizar productos para virtualizar actualizados con todos los parches publicados por su fabricante, en general, se suele estar cubierto de los ataques conocidos que explotan vulnerabilidades del producto. Por otro lado, para reducir las posibilidades de ataque en las empresas u organizaciones debido a malos diseños o errores en operación, se deben tener en cuenta las recomendaciones dadas por NIST y por ISO/IEC (comentadas más arriba), que además podrían mitigar problemas no parcheados de los productos utilizados. Ni en el trabajo del NIST ni en el de ISO/IEC se profundizan los temas de segmentación o aislamiento de red, mediante el uso de redes virtuales (virtual LANs o VLANS en inglés) y también quedaron fuera del alcance de este trabajo, pero es una herramienta que debe considerarse y tenerse en cuenta. Además, es muy importante observar las novedades que surgen relacionadas con seguridad en virtualización, para detectar si es necesario actualizar o cambiar las configuraciones de las herramientas utilizadas o los procedimientos aplicados en la operación de los productos.

Si se siguen las recomendaciones aquí presentadas, se evitan riesgos innecesarios al utilizar la virtualización. Creemos que esta herramienta es una tecnología madura y se deben aprovechar sus beneficios y ventajas.

9 Conclusión

El amplio uso de la virtualización y la relevancia que ha tomado gracias a la adopción de la computación en la nube hacen imprescindible que las áreas de seguridad informática tengan en cuenta los nuevos temas que surgen con esta tecnología. En este trabajo se han presentado distintos tipos de virtualización, haciendo foco en las MVs y se han comentado sus características principales. Se han presentado recomendaciones y guías para abordar el uso de esta herramienta, evitando cometer errores conocidos. Se ha mostrado una gran variedad de vulnerabilidades y ataques que han surgido con la virtualización.

Si bien son indudables las ventajas de la virtualización, también es cierto que se incorpora una nueva capa a gestionar, se agregan nuevas vulnerabilidades y nuevos tipos de ataque que hay que tener en cuenta. Aquí se han repasado los principales de ellos, comentando los temas que hay que tener en cuenta para reducir las probabilidades de éxito de los ataques.

Glosario

Escape de MV (*VM Escape*)
Es la acción de salir de la MV, ya sea para ganar acceso al hipervisor, al sistema operativo host o a otra MV.

Golden Images
Son las imágenes preconfiguradas o *templates* que se usan como base para crear MVs. También se la suele llamar imagen maestra o imagen base.

Hyperjacking
Es un tipo de escape de MV en el cual se toma control del hipervisor.

Hipervisor (*Hypervisor*)
Es la capa de software que permite la virtualización. La administra y monitorea, se ejecuta directamente sobre el *hardware* o bien sobre un sistema operativo. Es el que crea una capa de abstracción entre el *hardware* de la máquina física y el sistema operativo de la MV.

Imagen de MV (*VM Image*)
Es el archivo o directorio que contiene una MV. Incluye el sistema operativo, las aplicaciones, su configuración y los archivos de datos.

Jail break
Ver Escape de MV.

Máquina física (*Host machine*)
Es una máquina real donde corre la MV.

Máquina virtual (*Virtual machine*, MV)
Es una representación virtual de un sistema de computación, que mediante un programa provee un ambiente operativo que permite ejecutar programas como si fuera una computadora real.

Monitor de máquinas virtuales (*Virtual Machine Monitor*)
Ver hipervisor.

Sistema operativo invitado (*Guest operating system*)
Es el sistema operativo que corre sobre una MV. Generalizando se llama programa invitado (*software guest*) al programa que corre sobre una MV.

***Snapshot* (Instantánea)**
Es una captura de todo el estado de una MV en un determinado momento. Incluye sus discos, memorias, otros dispositivos, su estado (si está prendida, apagada), etc.

VM Guest Hopping
Es un tipo de escape de MV en el cual se salta desde una MV a otra.

VM Hopping
Ver *VM Guest Hopping*.

Anexos

Anexo A. Vulnerabilidades de Xen y VMware

A continuación, se presentan las vulnerabilidades reportadas para los productos de Xen y de VMware. Se presentan agrupados por año, separados por el tipo de vulnerabilidad. Antes de la tabla se explica brevemente en qué consiste cada vulnerabilidad. Se incluyeron los datos desde que salieron las primeras versiones de estos productos hasta fines de 2018, cuando se relevó esta información. Fuente: www.cvedetails.com.

Denial of Service (DoS): Ataque que busca consumir la mayor cantidad de recursos de un sistema para que los usuarios legítimos no puedan acceder al mismo.

Code Execution: Ataque que sube a servidores web archivos ejecutables maliciosos en sistemas que lo permiten.

Stack Overflow: Ocurre cuando una función copia datos en un buffer que exceden el tamaño de este último, sin que se valide y restrinja la cantidad de datos a copiar.

Corrupción de memoria: Son ataques donde se corrompe la memoria, con un *stack overflow*, corrupción del *heap* o *string format*. Deja el sistema inconsistente y puede colgar el equipo.

Cross-Site Scripting (XSS): En aplicaciones web un atacante puede ingresar texto en campos no validados, con el objetivo de robar información o alterar páginas.

Directory Traversal: A partir de parámetros de entrada no validados, un atacante puede ganar acceso no autorizado al sistema de archivos de un servidor.

HTTP response splitting: En este ataque se inserta un salto de línea en la cabecera de http para intentar tomar control del mismo.

Bypass Something: El atacante busca saltarse algún mecanismo de control para obtener más privilegios de los que le correspondan.

Gain Information: Esta vulnerabilidad permite obtener información que posteriormente puede ser usada para otro ataque.

Gain Privileges: Ocurre cuando el atacante logra obtener más privilegios que los inicialmente otorgados.

Cross Site Request Forgery (CSRF): En este ataque se busca que un usuario ejecute algún pedido sin ser consciente del mismo. Requiere que el usuario ya esté autenticado en el servicio que se intenta explotar.

Año	# de vulnerabilidades	DoS	Code Execution	Overflow	Corrupción de memoria	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF
1999	1			1								
2000	1	1										
2001	1											
2002	1		1	1								
2003	4										2	
2004	4	3	1									
2005	8	1	3	1		2					2	1
2006	6	1	1	1		1					2	
2007	25	11	5	4			1			2	5	
2008	31	6	5	6	2		2		1	3	10	
2009	20	7	5	4	1	1	2			1	3	
2010	24	2	6	2	1	4			1	1	7	
2011	18	5	3	2	1		2		1	2	3	
2012	34	10	7	6	1	4	3	1		4	11	1
2013	18	7	6	2	2		1		1		5	
2014	17	4	1			1				3	2	1
2015	15	8	4						1	1	2	
2016	36	8	6	5	4	6	2	1	2	3	8	1
2017	45	11	20	11		3			1	4	1	
2018	20	1	5	1		1			1	3		
Total	329	86	79	47	12	23	13	2	9	27	63	4
% del total		26,1	24	14,3	3,6	7	4	0,6	2.7	8.2	19.1	1.2

Tabla de vulnerabilidades de VMware por año y por tipo

Año	# de vulnerabilidades	DoS	Code Execution	Overflow	Corrupción de memoria	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF
2007	2											
2008	2		1	1								
2009	2	1										
2012	35	31	3	3	5					1	5	
2013	42	29	2	9	3					6	8	
2014	45	42	2	10	1					3	8	
2015	41	29	4	5	1					6	3	
2016	28	18	1	3						7	10	
2017	62	37	6	4	3					15	17	
2018	22	13	2	1					1	3	6	
Total	281	200	21	36	13				1	41	57	
% del total		71,2	7,5	12,8	4,6	0	0	0	0,4	14,6	20,3	0

Tabla de vulnerabilidades de Xen por año y por tipo

Anexo B. Canal encubierto usando la *cache*

A continuación, se presenta en pseudo-código el algoritmo clásico usado para un canal encubierto mediante el uso de la *cache*. Si bien cada implementación varía en los detalles, la presentada aquí es la idea básica.

Cache[*N*]: La *cache* se divide en *N* regiones. Cada región puede estar en uno de los siguientes estados: *cached* o *flushed.*

DSend[*N*]: *N* es el *bit* que se envía
DRecv[*N*]]: *N* es el *bit* que se recibe

Sender Operations:	**Receiver Operations:**

(Wait for receiver to initialize the cache)	**for** $i := 0$ **to** $N-1$ **do**
	{Put $Cache[i]$ into the *cached* state}
	Access memory maps to $Cache[i]$;
	end for

for $i := 0$ **to** $N-1$ **do**
if $D_{Send}[i] = 1$ **then**
{Put $Cache[i]$ into the *flushed* state}
Access memory maps to $Cache[i]$;
end if
end for
	(Wait for sender to prepare the cache)

(Wait for receiver to read the cache)	**for** $i := 0$ **to** $N-1$ **do**
	Timed access memory maps to $Cache[i]$;
	{Detect the state of $Cache[i]$ by latency}
	if $AccessTime > Threshold$ **then**
	$D_{Recv}[i] := 1$; {$Cache[i]$ is *flushed*}
	else
	$D_{Recv}[i] := 0$; {$Cache[i]$ is *cached*}
	end if
	end for

Figura 9. Algoritmo para canal encubierto usando la cache. Fuente: [60]

Bibliografía

[1] W. Brand, Virtualization for dummies, 3rd Stratus Special Edition, New Jersey: John Wiley & Sons, Inc., 2015.

[2] P. Tsai, «Server Virtualization and OS Trends» 2016. [En línea]. Available: https://community.spiceworks.com/networking/articles/2462-server-virtualization-and-os-trends. [Último acceso: 01 01 2019].

[3] H. Singh y M. Yip, Next-Gen Virtualization, New Jersey, USA: John Wiley & Sons, Inc., 2017.

[4] F. Bazargan , C. Yeun y J. Zemerly, «State-of-the-Art of Virtualization, its Security Threats and Deployment Models» *International Journal for Information Security Research,* 2013.

[5] P. Dawson y N. Hill, «Hype Cycle for Virtualization» *Gartner,* 2013.

[6] A. Khreishah, M. Azeem y I. Khalil, «Cloud Computing Security: A Survey» *Computers,* nº 3, pp. 1-35, 2014.

[7] K. Scarfone , M. Souppaya y P. Hoffman, «Guide to Security for Full Virtualization Technologies, Recommendations of the National Institute of Standards and Technology» *NIST,* 2011.

[8] H. Yin, P. Poosankam, S. Hanna y D. Song, «Detection of Intrusions and Malware, and Vulnerability Assessment» Springer, Bonn, Alemania,, 7th International Conference, 2010.

[9] Y. Zhang, A. Juels, M. Reiter y T. Ristenpart, «Cross-VM side channels and their use to extract private keys» de *The 2012 ACM Conference on Computer and Communications Security*, 2012, pp. 305-316.

[10] T. Ristenpart, E. Tromer, H. Shacham y S. Savage, «Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds» de *The ACM Conference on Computer and Communications Security*, 2009, pp. 199-212.

[11] A. Jasti, P. Shah, R. Nagaraj y R. Pendse, «Security in multi-tenancy cloud» de *44th Annual IEEE International Carnahan Conference on Security Technology*, San Jose, CA, USA, 2010.

[12] S. D'Antonio, L. Coppolino, G. Mazzeo y L. Romano, «Cloud security: Emerging threats and current solutions» *Computers and Electrical Engineering,* Nápoles, Italia, pp. 1-15, 2016.

[13] V. Varadarajan, Y. Zhang, T. Ristenpart, C. Tech y M. Swift, «A Placement Vulnerability Study in Multi-Tenant Public Clouds» *24th USENIX Conference on Security Symposium,* Washington, USA, pp. 913-928, 2015.

[14] B. Golden, Virtualization for dummies,1st Edition, Wiley Publishing, 2007.

[15] B. Bitner y S. Greenlee, z/VM – A Brief Review of Its 40 Year History, IBM Corporation, 2012.

[16] IBM, «5749-010 - VIRTUAL MACHINE FACILITY/370 (VM/370)» [En línea]. Available: http://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_sm/0/897/ENUS5749-010/index.html&lang=en&request_locale=en. [Último acceso: 11 03 2019].

[17] G. Popek y R. Goldberg, «Formal requirements for virtualizable third generation architectures» *Communications of the ACM,* vol. 17, nº 7, p. 412–421, 1974.

[18] K. Adams, K. yAgesen O. y O. Agesen, «A Comparison of Software and Hardware Techniques for x86 Virtualization» San Jose, California, VMware, ASPLOS, 2006.

[19] F. Haro, F. Freitag, L. Navarro, E. Sánchez, N. Mendoza, J. Guerrero-Ibañez y A. González, «A summary of virtualization techniques» Procedia Technology, 2012.

[20] Intel, Intel® Virtualization Technology Specification for the IA-32 Intel® Architecture, 2005.

[21] AMD, AMD64 Virtualization Codenamed "Pacifica" Technology: Secure Virtual Machine Architecture Reference Manual, 2005.

[22] B. Sodhi, Topics in Virtualization and Cloud Computing, 2017.

[23] S. Luan, Exploit Two Xen Hypervisor Vulnerabilities, Alibaba Cloud, 2016.

[24] VMware, Virtualization Overview, White Paper, California, USA: VMware Inc., 2006.

[25] W. Chen, J. Chan, O. Mueller, M. Singh y T. Väättänen, «DB2 Virtualization» de *RedBooks*, IBM, 2009.

[26] M. Tulloch, Understanding Microsoft Virtualization Solutions, ebook: Microsoft, 2010.

[27] VMware, Software and Hardware Techniques for x86 Virtualization, VMware Inc., 2009.

[28] A. Litchfield y A. Shahzad, A Systematic Review of Vulnerabilities in Hypervisors and Their Detection, AMCIS, 2017.

[29] ISO/IEC, ISO/IEC FDIS 21878, Information technology - Security techniques - Security guidelines for design and implementation of virtualized servers, Final Draft, Suiza, 2018.

[30] M. Noorafiza, H. Maeda, T. Kinoshita y R. Uda, «Virtual Machines Detection Methods Using Ip Timestamps Pattern Characteristic» *International Journal of Computer Science & Information Technology (IJCSIT),* vol. 8, nº 1, 2016.

[31] A. Huseinovic y S. Ribic, «Virtual machine memory forensics» Belgrado, Serbia, 21st Telecommunications Forum Telfor, 2013.

[32] S. Dhawale, «Virtualization security in Data Centres & cloud» *International Journal of Scientific & Engineering Research,* vol. 5, nº 1, 2014.

[33] P. Ferrie, Attacks on Virtual Machine Emulators, Symantec Advanced Threat Research, 2007.

[34] P. Chen y B. Noble, When Virtual is Better than Real, Washington, USA: HotOS, 2001.

[35] A. Gkortzis, S. Rizou y D. Spinellis, «An empirical analysis of vulnerabilities in virtualization technologies» *IEEE,* pp. 533-538, 2016.

[36] L. Miller, Server Virtualization for Dummies, Oracle Special Editioin, New Jersey: John Wiley & Sons, Inc., 2012.

[37] M. Lococo, «Virtualization and Security Boundaries» 2009. [En línea]. Available: http://mikelococo.com/files/2009/virtualization-and-security-boundaries.pdf. [Último acceso: 15 01 2019].

[38] D. Vile, T. Lock, M. Atherton y J. Collins, Desktop Virtualization for Dummies, West Sussex, England: John Wiley & Sons, 2010.

[39] W. Wong, «What's the Difference Between Containers and Virtual Machines?» 2016. [En línea]. Available: https://www.electronicdesign.com/dev-tools/what-s-difference-between-containers-and-virtual-machines. [Último acceso: 17 03 2019].

[40] J. Brodkin, «Virtual server sprawl highlights security concerns» Network World, 2008. [En línea]. Available: https://www.networkworld.com/article/2278830/data-

center/virtual-server-sprawl-highlights-security-concerns.html. [Último acceso: 31 12 2018].

[41] P. Tsai, «Hypervisor Market Share – ControlUp Perspective» 2018. [En línea]. Available: https://www.controlup.com/hypervisor-market-share-controlup-perspective. [Último acceso: 01 01 2019].

[42] CoreLabs, «Path Traversal vulnerability in VMware's shared folders implementation» 2008. [En línea]. Available: https://www.secureauth.com/labs/advisories/advisory-vmware. [Último acceso: 31 12 2018].

[43] K. Kortchinsky, Cloudburst, Black hat: Immunity, 2009.

[44] R. Randel, Virtualization Security and Best Practices, VMware, 2006.

[45] Cloud Security Alliance, The Notorious Nine: Cloud Computing Top Threats in 2013, Top Threats Working Group, 2013.

[46] Cloud Security Alliance, The Treacherous 12, Top Threats to Cloud Computing, CSA, 2017.

[47] T. Ritter, Virtualization Security, Achieving Compliance for the Virtual Infrastructure, Nemertes Research, 2009.

[48] S. Dhawale, «Virtualization security in Data Centres & cloud» *International Journal of Scientific & Engineering Research,* vol. 5, nº 1, 2014.

[49] Cloud Security Alliance, Best Practices for Mitigating Risks in Virtualized Environments, CSA, 2015.

[50] A. Murphy, Security Implications of the Virtualized Datacenter, F5 White Paper, 2007.

[51] J. Oberheide, E. Cooke y F. Jahanian, «Empirical exploitation of live virtual machine migration» de *Electrical Engineering and Computer Science Department*, Michigan, USA, University of Michigan, 2008.

[52] D. Shackleford , Virtualization Security: Protecting Virtualized Environments, Indiana, EE.UU.: Sybex, 2012.

[53] S. King, P. Chen, Y. Wang, C. Verbowski, H. Wang y J. Lorch, «SubVirt: Implementing malware with virtual machines» de *Conference IEEE Symposium on Security and Privacy*, Berkeley, California, USA, 2006.

[54] J. Rutkowska, «The Invisible Things Lab's blog, Introducing Blue Pill» 2006. [En línea]. Available: http://theinvisiblethings.blogspot.com.ar/2006/06/introducing-blue-pill.html. [Último acceso: 15 01 2019].

[55] J. Rutkowska, «Is Game Over() Anyone?» Black Hat, 2007. [En línea]. Available: https://www.blackhat.com/presentations/bh-usa-07/Rutkowska/Presentation/bh-usa-07-rutkowska.pdf. [Último acceso: 15 01 2019].

[56] Z. Wang y R. Lee, «Covert and Side Channels due to Processor Architecture» de *22nd Annual Computer Security Applications Conference*, Florida USA, 2006, pp. 473-482.

[57] J. Betz, D. Westhoff y G. Müller, Survey on covert channels in virtual machines and cloud computing, Trans Emerging Tel Tech, 2016.

[58] K. Okamura y Y. Oyama, Load-based covert channels between Xen virtual machines, Proceedings of the 2010 ACM Symposium on Applied Computing, 2010.

[59] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen y R. Schlichting, «An Exploration of L2 Cache Covert Channels in Virtualized Environments» de *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, Chicago, Illinois, USA, 2010, pp. 29-40.

[60] Z. Wu, Z. Xu y H. Wang, «Whispers in the Hyper-Space: High-Speed Covert Channel Attacks in the Cloud» de *Proceedings of the 21st USENIX Conference on Security Symposium*, Bellevue, Washington, USA, 2012, pp. 159-173.

[61] C. Maurice, C. Neumann, O. Heen y A. Francillon, «C5: Cross-Cores Cache Covert Channel» de *Detection of Intrusions and Malware, and Vulnerability Assessment: 12th International Conference*, Milan, Italy, 2015, pp. 46-64.

[62] Y. Younis, K. Kifayat y M. Merabti, «Cache Side-Channel Attacks in Cloud Computing» de *2nd International Conference on Cloud Security Management*, 2014.

[63] H. Hlavacs, T. Treutner, J. Gelas, L. Lefevre y A. Orgerie, «Energy Consumption Side-Channel Attack at Virtual Machines in a Cloud» *IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing,* 2011.

[64] G. Irazoqui, M. Inci, T. Eisenbarth y B. Sunar, Wait a Minute! A fast, Cross-VM Attack on AES, Gothenburg, 2014.

[65] S. Anwar, Z. Inayat, M. Zolkipli, J. Zain, A. Gani, N. Anuar, M. Khan y V. Chang, «Cross-VM cache-based side channel attacks and proposed prevention mechanisms: A survey» *Journal of Network and Computer Applications,* vol. 93, pp. 259-279, 2017.

[66] C. Rebeiro, D. Mukhopadhyay y S. Bhattacharya, «Time-Driven Cache Attacks» de *En Timing Channels in Cryptography*, Cham, Suiza, Springer, 2015, pp. 53-70.

[67] J. Rutkowska, «Red Pill... or how to detect VMM using (almost) one CPU instruction» 2004. [En línea]. Available: https://web.archive.org/web/20070911024318/http://invisiblethings.org/papers/redpill.html. [Último acceso: 01 22 2019].

[68] J. Franklin, M. Luk, J. McCune, A. Seshadri, A. Perri y L. Van Doorn, «Remote detection of virtual machine monitors with fuzzy benchmarking» *ACM SIGOPS Operating Systems Review,* nº 42, pp. 83-92, 2008.

[69] C. Jämthagen, On Offensive and Defensive Methods in Software Security, Lund University, 2016.

[70] L. Coppolino, S. D'Antonio, G. Mazzeo y L. Romano, «Cloud security: Emerging threats and current solutions» de *Computers and Electrical Engineering*, Nápoles, Italia, 2016, pp. 1-15.

[71] P. Di Tommaso, E. Palumbo, M. Chatzou, P. Barja, M. L Heuer y C. Notredame, «The impact of Docker containers on the performance of genomic pipelines» *PeerJ,* 2015.