Capítulo 3

"*User Mode Linux*"

3.1. Características de *User Mode Linux*

User Mode Linux (UML) es diferente a cualquier otra herramienta de virtualización libre o comercial que se ejecute sobre Linux. El hecho de que fuera adoptado como la única herramienta de virtualización que forma parte del *kernel* Linux (a partir de la versión 2.6.10) hizo que este *software* fuera incrementando la popularidad y desplazando a otros productos de virtualización.

La diferencia entre UML y otros *softwares* de virtualización es que UML implementa máquinas virtuales a partir de la virtualización de un sistema operativo. Esta filosofía de diseño lo hace particular respecto de otras herramientas que compiten en el área considerada. Algunas de las características más relevantes de UML son:

- Los diseñadores de UML advirtieron que emular la arquitectura de un sistema basado en microprocesador consume demasiados recursos comparado con la virtualización de un SO. Esta característica posibilita que UML tenga medidas de desempeño comparables respecto de otras herramientas de virtualización. Como un dato descriptivo y considerando cualquier aplicación, UML la ejecuta con un retardo que en el peor de los casos es un 40% mayor respecto a sí la misma aplicación fuese ejecutada directamente en el sistema anfitrión.

- Cuenta con un foro que realimenta al grupo desarrollo de UML. Como consecuencia UML es constantemente actualizado y renovado para soportar las futuras exigencias.

- El *kernel* de UML es independiente del *kernel* anfitrión y más aún del hardware.

- Cuenta con más de cinco años de desarrollo y constantes avances e innovaciones como por ejemplo el soporte de máquinas virtuales anidadas. o UML sobre UML.

- Es un *software* cuyo código es libre y gratuito. Se distribuye bajo la licencia GPL (Licencia General Pública).

3.2 Áreas de Aplicación

UML es una poderosa herramienta que puede ser aplicada para múltiples propósitos [1]. Algunos de ellos son:

1. Experimentar con la última versión de kernel de manera segura.

2. Experimentar con distintas distribuciones simultáneamente, sin tener que configurar particiones de disco. La simultaneidad con diferentes distribuciones de Linux no es un problema.

3. Experimentar con el código fuente del *kernel* (Guest) en forma segura. realizando modificaciones sin provocar desastres que por lo general generan reinstalaciones. Desarrollo experimental.

4. Experimentar con topologías de red y sus protocolos asociados. Es ideal como herramienta de pruebas en lo que se refiere a redes de computadoras y sistemas operativos.

5. Ideal para aplicarlo en ambientes de formación de recursos humanos (Educación).

6. Experimentación de aplicaciones utilizando el concepto de *sandbox* o caja segura.

7. Es posible aplicarlo para la creación de clusters virtuales

8. *Honeynet*

9. Complejos escenarios de red que involucren múltiples dispositivos se pueden generar y ejecutar en PC.

Hasta aquí se describió someramente algunas de las áreas de aplicación de UML. A continuación se abordará la misma temática en mayor detalle teniendo en cuenta que puede significar nuevas áreas de investigación en un futuro cercano.

3.2.1 Experimentando con Diferentes Versiones de *Kernels*.

Las últimas versiones de *kernel*s pueden tener vulnerabilidades. En este escenario UML constituye una herramienta ideal para experimentar con distintas aplicaciones y detectar fallas sin alterar el funcionamiento del sistema operativo anfitrión.

3.2.2 Experimentando con Distribuciones.

UML permite ejecutar distintas distribuciones de Linux sin necesidad de particionar el disco e instalar en una máquina “real”. De esta forma se puede experimentar con distintas distribuciones de las muchas existentes y así determinar las ventajas y desventajas de cada una de ellas.

3.2.3 Desarrollo Experimental.

Realizar modificaciones al código fuente del *kernel* puede ser una tarea arriesgada, si no se cuenta con los conocimientos necesarios. Es por ello que UML se convierte en una herramienta importante para aquellos que están interesados en experimentar con el código fuente, así como también con las distintas configuraciones del *kernel* Linux.

3.2.4 Como Entorno para Pruebas.

Las máquinas virtuales basadas en UML constituyen una herramienta ideal como entorno de pruebas. En ellas es posible realizar todo tipo de pruebas que de otro modo serían sumamente "riesgosas" y pondrían en peligro la integridad del sistema anfitrión. Por ejemplo se puede ejecutar un comando desde consola rm / -rf sin temor a ocasionar la destrucción de archivos en el *host* anfitrión.

3.2.5 Educación.

Los escenarios de red basados en UML permiten vislumbrar nuevas oportunidades en el campo de las tecnologías de comunicaciones basadas en IP. Algunos protocolos de comunicación son bastante complejos, especialmente los que involucran aspectos de seguridad, donde la teoría es escasa y la práctica se convierte en algo esencial. UML es una herramienta de gran importancia para los estudiantes a la hora de aprender mediante la experiencia. Cuando el problema es presupuestario, UML es ideal para enseñar todo lo relativo a SO Linux, como así también administración de redes y administración general de sistemas.

3.2.6 Como una Caja Segura

Por defecto los procesos que se ejecutan dentro de UML no tienen acceso al sistema operativo de la máquina anfitriona, por lo tanto los programas maliciosos que se encuentren dentro de UML no pueden dañar el sistema real.

3.2.7 Clusters de Computadoras

Un cluster es un tipo particular de computadora paralela, es decir, un conjunto de computadoras que pueden trabajar de manera coordinada en la solución de un mismo problema. Aunque no existe un acuerdo general en cuanto a la definición exacta de lo que significa un cluster, y muchas computadoras diferentes pueden llegar a ser clasificadas como tales, dos de las características más aceptadas de estos equipos y quizás las que más han influido en su rápido desarrollo son que se construyen a partir de componentes que pueden encontrarse en el mercado común de cómputo, lo que en EEUU se conoce como "*commodity of the shelf*" (COTS), y que se desarrollan bajo el esquema de "hágalo usted mismo". Hoy en día, tienen un papel importante en la solución de problemas de las ciencias, las ingenierías y aplicaciones comerciales.

Los clusters han evolucionado para apoyar actividades en aplicaciones que van desde súper computo y *software* de misiones críticas, servidores Web y comercio electrónico, bases de datos de alto rendimiento, etc. [17].

El procesamiento en clusters surge como resultado de la convergencia de varias tendencias que incluyen, la disponibilidad de microprocesadores de alto rendimiento más económicos y redes de alta velocidad, el desarrollo de herramientas de *software* para el procesamiento distribuido de alto rendimiento, y la creciente necesidad de potencia computacional para

aplicaciones en las ciencias computacionales y comerciales.

Por otro lado, la evolución y estabilidad que ha alcanzado el sistema operativo Linux, ha contribuido desarrollo de muchas tecnologías nuevas, entre ellas la de clusters.

Mucha gente pensaría que las palabras "cluster" o "cluster de servidores" indican un grupo de computadoras de alto rendimiento utilizados en las investigaciones científicas. Sin embargo este es sólo un tipo de cluster. La idea detrás del concepto de "clusters de alto rendimiento" es hacer que un número grande de máquinas individuales actúen como una sola máquina muy potente. Este tipo de clusters se aplica mejor en problemas grandes y complejos que requieren una cantidad enorme de potencia computacional. Entre las aplicaciones más comunes de clusters de alto rendimiento se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes, etc.

Un segundo tipo de tecnología de clusters, es el "clusters de servidores virtuales", permite que un conjunto de servidores de red compartan la carga de balance de tráfico de sus clientes. Al balancear la carga de trabajo de tráfico en un arreglo de servidores, mejora el tiempo de acceso y confiabilidad. Además como es un conjunto de servidores el que atiende el trabajo, la falla de uno de ellos no ocasiona una falla en el sistema. Este tipo de servicio es de gran valor para compañías con altos volúmenes de tráfico en sus sitios Web.

El último tipo importante de clusters, involucra el tener servidores que actúan entre ellos como respaldos de la información que entregan. Este tipo de clusters se les conoce como "clusters de alta disponibilidad" o "cluster de redundancia".

UML puede ser utilizado para implementar cualquiera de los tipos de clusters anteriormente mencionados sin ningún tipo de problema. En la actualidad existen numerosos proyectos de *clustering* que involucran a UML en su desarrollo[31].

3.2.8 Honeynet.

Las *honeynet* son un tipo de *honeypot* o máquinas expuestas a internet para que los hackers “jueguen” con ellas. En general, se utilizan para aprender acerca de los comportamientos y las nuevas técnicas que usan los intrusos informáticos. Un *honeypot* es un recurso cuyo valor se basa en ser analizado, atacado o comprometido. Una *honeynet* es un *honeypot* con gran interacción, es decir se ofrecen sistemas operativos reales para que los atacantes interactúen con él [6].

Las *honeynet* virtuales toman el concepto de las tecnologías de las *honeynet*, y las implementan en un único sistema. Esta implementación tiene sus ventajas y desventajas comparadas con las *honeynet* tradicionales. Las ventajas son costo reducido y fácil manejo, ya que todo está combinado en un único *host*. Sin embargo, esta simplicidad tiene asociada la limitación relacionada con los tipos de sistemas operativos que pueden implantarse debido al hardware y al programa virtual.

Tradicionalmente, la información sobre seguridad ha sido puramente defensiva. Cortafuegos, Sistemas de Detección de Intrusiones, cifrado, todos estos mecanismos se usan defensivamente para proteger los recursos de alguien. La estrategia es defender una organización tan bien como sea posible. Las *honeynet* intentan cambiar este concepto. El principal propósito de una *honeynet* es recoger información sobre las amenazas existentes en base a una exposición verdadera del producto a defender. De este modo y luego del análisis de la información registrada es posible determinar soluciones y así mejorar las vulnerabilidades de un determinado sistema operativo. Las *honeynet* no son más que una herramienta, y como tal puede ser usada para otros fines. Un ejemplo, las organizaciones

pueden utilizar *honeynet* para comprobar y desarrollar su capacidad de respuesta ante incidentes. Conceptualmente, las *honeynet* son un mecanismo simple. Se crea una red donde se puede ver todo lo que ocurre dentro de ella. Esto permitirá observar a los *hackers* interactuar con su entorno virtual. Esta red controlada se convierte en una *honeynet*. Las actividades capturadas enseñan las herramientas, tácticas y motivos de la comunidad de *hackers*.

Tradicionalmente, el gran problema de los profesionales de la seguridad reside en que la detección y captura de actividad *hacker* supone una sobrecarga de información. El reto para muchas organizaciones es determinar de una enorme cantidad de información qué es tráfico productivo y qué es actividad maliciosa. Herramientas y técnicas como los Sistemas de Detección de Intrusiones, análisis forenses de las máquinas, o los análisis de los registros del sistema intentan resolver esto mediante una base de datos de marcas conocidas o algoritmos para determinar qué es tráfico de producción y qué es actividad maliciosa. Sin embargo, la sobrecarga de información, la contaminación de los datos, actividades no descubiertas puede hacer el análisis y la determinación de las actividades algo extremadamente difícil.

Como todos los *honeypot*s, las *honeynet* resuelven este problema de la sobrecarga de información a través de la simplicidad. Una *honeynet* es una red diseñada para ser comprometida, no para ser usada por tráfico productivo. Así, cualquier tráfico entrando o saliendo de la red es sospechoso por definición. Cualquier conexión iniciada desde fuera de la *honeynet* hacia dentro de la red es probablemente algún tipo de sondeo, ataque u otra actividad maliciosa. Cualquier conexión iniciada desde dentro de la *honeynet* hacia otra red de afuera indica que un sistema fue comprometido. Un agresor ha iniciado una conexión desde su recién atacada computadora y ahora está saliendo a Internet. Este concepto de ningún tráfico productivo simplifica enormemente la captura de datos y el análisis.

3.2.9 Experimentación en Redes.

UML además de proporcionar la manera de crear múltiples máquinas virtuales, también implementa mecanismos para su interconexión formando redes virtuales. Desde el punto de vista de la máquina virtual, se utiliza una interfaz de red (virtual) de forma transparente. Este tópico será abordado más detalladamente en capítulos posteriores.

Los usos de los escenarios virtuales son prácticamente los mismos que los de los escenarios de red reales. Las topologías de red creadas con UML permiten realizar pruebas de aplicaciones de red (tales como servidores *web*, DNS, *mail*,etc), especialmente cuando dichas pruebas están orientadas a la funcionalidad de herramientas más que a medidas de eficiencia.

Los escenarios de red pueden ser utilizados para el estudio en profundidad de protocolos de red, brindando la posibilidad de:

- Aplicar los conocimientos teóricos sobre protocolos TCP/IP adquiridos en la configuración de una red IP que puede estar compuesta por *router*s, *switch*s, *hub*s, *gateway*, varios segmentos de LAN y varios sistemas finales (como clientes).
- Comprender el funcionamiento de algunos protocolos de la arquitectura TCP/IP, como IP, ICMP o ARP, y de algunas herramientas de diagnóstico como *ping* y *traceroute*, todo ello mediante la realización de pruebas sobre un escenario de red virtual.

- Familiarizarse con el manejo de un analizador de protocolos como herramienta de diagnóstico de redes, como el *ethereal*, *tcpdump*, etc.
- Comprender el funcionamiento y configuración del encaminamiento estático y dinámico en una red IP.