ESCUELA POLITÉCNICA DEL EJÉRCITO

VICERRECTORADO DE INVESTIGACIÓN Y VINCULACIÓN CON LA COLECTIVIDAD

UNIDAD DE GESTIÓN DE POSTGRADOS

MAESTRÍA EN REDES DE LA INFORMACIÓN Y CONECTIVIDAD

REPOTENCIACIÓN DE UN SISTEMA DE FIREWALL DE CÓDIGO ABIERTO BASADO EN FUNCIONALIDADES DE PLATAFORMA PROPIETARIA

Tesis de grado

Patricio Xavier Zambrano Rodríguez

Marco Polo Sánchez Aguayo

Sangolquí, 2013

DECLARACIÓN

Nosotros, Marco Polo Sánchez Aguayo y Patricio Xavier Zambrano Rodríguez, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que se ha consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica del Ejército, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por los Ingenieros Marco Polo Sánchez Aguayo y Patricio Xavier Zambrano Rodríguez bajo mi dirección.

_____

Dr. Walter Fuertes, PhD.

Director de Tesis

AUTORIZACION

Yo, Marco Polo Sánchez Aguayo, autorizo a la Escuela Politécnica del Ejército, publicar la tesis que tiene como título "REPOTENCIACIÓN DE UN SISTEMA DE FIREWALL DE CÓDIGO ABIERTO BASADO EN FUNCIONALIDADES DE PLATAFORMA PROPIETARIA", en el repositorio público de la ESPE.

_____

Ing. Marco Polo Sánchez Aguayo

AUTORIZACION

Yo, Patricio Xavier Zambrano Rodríguez, autorizo a la Escuela Politécnica del Ejército, publicar la tesis que tiene como título "REPOTENCIACIÓN DE UN SISTEMA DE FIREWALL DE CÓDIGO ABIERTO BASADO EN FUNCIONALIDADES DE PLATAFORMA PROPIETARIA", en el repositorio público de la ESPE.

_____

Ing. Patricio Xavier Zambrano Rodríguez

Dedicatoria

A mi esposa e hija.

Dedico este trabajo, por haberme impulsado y apoyado incondicionalmente hasta este momento tan importante de mi formación profesional.

A mis padres y hermanas.

Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien.

Patricio Zambrano

Dedico el desarrollo de esta Tesis a toda mi familia.

Para mi esposa Paulina, a ella especialmente dedico esta Tesis. Por su paciencia, comprensión, empeño, fuerza, por su amor, por ser tal y como es. Es la persona que más directamente ha sufrido las consecuencias del trabajo realizado. Realmente ella me llena por dentro para conseguir un equilibrio que me permita dar el máximo de mí. Nunca le podré estar suficientemente agradecido.

Para mis hijos, Matías y Julián. Su nacimiento coincidió con el desarrollo de la Tesis. Es lo mejor que me ha pasado, y han venido a este mundo para darme el último empujón para terminar este trabajo. Es sin duda mi referencia para el presente y para el futuro.

Marco Sánchez

Agradecimientos

Primero y como más importante, nos gustaría agradecer sinceramente a nuestro director y tutor de Tesis, Dr. Walter Fuertes, su esfuerzo y dedicación. Sus conocimientos, orientaciones, su manera de trabajar, persistencia, paciencia y motivación han sido fundamentales para nuestra formación como investigadores. Él ha inculcado en nosotros un sentido de seriedad, responsabilidad y rigor académico sin los cuales no podríamos tener una formación completa como investigadores. A su manera, ha sido capaz de ganarse nuestra lealtad y admiración, así como sentirnos en deuda con él por todo lo recibido durante el periodo de tiempo que ha durado esta Tesis.

Resumen

En la actualidad los avances significativos que ha experimentado el campo de la seguridad informática ha permitido establecer mecanismos más rígidos de control, pero en su contra parte también han existido avances en relación a las alternativas de ataques tanto a redes públicas como privadas, al punto en el que se puede afirmar que no existen redes seguras sino únicamente redes confiables. En este contexto una red confiable responderá de acuerdo a las políticas de seguridad establecidas y que esta no sea fácilmente vulnerada.

Este proyecto tiene establecido el evaluar y repotenciar una plataforma de FIREWALLING de código abierto basado en las características distintivas de un sistema propietario y verificar su comportamiento en términos de performance y efectividad ante ataques informáticos, con el fin reforzar la detección, control y mitigación de ataques de seguridades informáticos en las instituciones gubernamentales. En este sentido se realizó un análisis de varios Firewalls Open Source, para lo cual se estableció como método de selección evaluar parámetros como Sistema Operativo y Soporte a las distribuciones, determinando luego de este análisis como herramientas seleccionadas a ClearOS, Zentyal y Pfsense.

Una vez conformado el ambiente experimental se establecieron los procedimientos y los tipos de ataques para la ejecución de pruebas, orientadas a la evaluación del desempeño, consumo de CPU, memoria, de los sistemas de FIREWALL. El esquema de pruebas para calcular el consumo de recursos de CPU y Memoria del FIREWALL, fue establecido en base parámetros previamente analizados: Nº de páginas Web solicitadas; tiempo de duración de la prueba y periodo de solicitud de páginas Web.

Las pruebas fueron efectuadas desde un equipo de la red local, el mismo que en base a la generación de una secuencia de comandos (algoritmo) de procesamiento en bach, generaba la solicitud de múltiples conexiones hacia páginas Web.

En relación a la evaluación de los resultados se puedo apreciar, que el desempeño de cada uno de los firewalls analizados, es directamente proporcional al nivel de aplicabilidad que se desee de los mismos. Dentro de este análisis se pudo determinar que ClearOS (SO RedHat) obtuvo resultados muy similares a Zentyal (SO Debian) en el uso de los recursos de Hardware, sin embargo la optimización de memoria RAM, soporte técnico del sistema operativo y de aplicación y mejor desempeño ante actividades sospechosas de RED, permite seleccionar a esta herramienta para su posterior repotenciación en aspectos de seguridad.

Una vez seleccionada la distribución (CelarOs), se realizó la repotenciación de la misma estableciendo la implementación de mecanismos de mejora como la adición de las herramientas IPS (PSAD), TCPDUMP, P0F, NESSUS y la modificación del código fuente de ClearOS, para que desde la misma GUI de esta distribución administrar dichas herramientas.

ÍNDICE

Pag.

Lista de Figuras

Lista de tablas

CAPITULO I

INTRODUCCIÓN

Los sistemas de información de las instituciones sufren permanentemente ataques cibernéticos, a pesar de los esfuerzos en implementar y adquirir mecanismos de detección y control como los firewall. Los ataques cibernéticos se han incrementado en los últimos años, y uno de sus destinos son las entidades gubernamentales a nivel mundial (Tecnología, 2012), motivo por el cual los investigadores están dedicando grandes esfuerzos para detectarlos y mitigarlos.

Uno de los mecanismos de detección más robustos técnicamente, son los firewalls. Estos dispositivos constituyen la defensa principal o de frontera en la seguridad de las redes contra los ataques y tráfico no autorizado. Sin embargo, la eficacia de estos mecanismos es dependiente de las técnicas de gestión de políticas que los administradores de red pueden utilizar para analizar, depurar y verificar la corrección de las reglas de filtrado (Al-Shaer, 2005).

En la actualidad, debido a la evolución y sofisticación de los ataques, la industria ha diseñado múltiples herramientas de software de código abierto (Rash, 2007), incorporando varios niveles de seguridad (Lau, 2000). También ha desarrollado diversas técnicas de mejoramiento en la detección y control de ataques que tiene los firewalls (G. Misherghi, 2008) (Ying-Dar Lin, 2010) (Sheth, 2011) (S. Patton, 2000) e inclusive ha reforzado las decisiones de firewall con técnicas de aprendizaje automático como las huellas dactilares, para inferir la solución (Hulst, 2012). En este mismo contexto, la comunidad científica ha realizado estudios comparativos de la efectividad de los firewalls (Khaled Salah, 2012) (JeeHyun Hwang, 2012) (H. Hu, 2012); no

obstante, el número de vulnerabilidades también está en crecimiento, según lo señalan las estadísticas publicadas por la *National Vulnerability Database*, que indican que en el tercer trimestre de 2011, el número de vulnerabilidades registradas fue de aproximadamente 50.000.

Ante este problema, esta investigación, se enfoca en evaluar y repotenciar una plataforma de firewall de código abierto basado en las características distintivas de un sistema propietario y verificar su comportamiento en términos de performance y efectividad ante ataques informáticos, con el fin reforzar la detección, control y mitigación de ataques de seguridades informáticos en las empresas.

Para llevarlo a cabo se diseñó e implementó un escenario de red experimental LAN/WAN separado por un dispositivo firewall configurado en tres diversas implementaciones de software: Pfsense (Pfsense, 2012), Zential (Zentyal, 2012) y Clear OS (OS, 2012), alternándolos en tres distribuciones distintas de Linux, Redhat, Debian y FreeBSD respectivamente, pero sobre el mismo entorno real que permitió efectuar las diferentes pruebas a los firewalls seleccionados, aplicando un esquema similar de pruebas para una justa comparación.

Como principales contribuciones de este trabajo de investigación se puede mencionar: i) la evaluación cuantitativa de tres firewalls de código abierto; ii) la generación de un ambiente de pruebas real en una sola máquina, mediante la simulación de la solicitud de un centenar de páginas Web, resolviendo la limitación del número de equipos físicos requeridos; y, iii) la repotenciación de un firewall de código abierto, modificando su interfaz gráfica de usuario, reprogramando el código fuente, e incorporando librerías para disponer de una solución integral y consolidada de varios servicios de seguridad.

1.1 JUSTIFICACIÓN

Este proyecto pretende dar respuesta a un problema actual y real que afecta a los servicios y prestaciones de las redes de información. Así mismo, dada la obligatoriedad de acogerse al mandato constitucional 1014, este proyecto, favorece la posibilidad de evaluar sí el firewall de software libre, es lo suficientemente robusto y tiene la efectividad requerida.

Desde el punto de vista de gestión, este trabajo, permitirá plantear a los administradores de red, procedimientos, métricas o parámetros técnicos a la hora de securizar un entorno de red. Adicionalmente, desde el punto de vista gubernamental, la implementación del presente proyecto, permitiría reducir costos de licenciamiento en software o en inversión de hardware. Finalmente, desde el punto de vista investigativo, esta exploración intentará repotenciar un sistema de firewalling, mediante el mejoramiento de su código fuente, generando así conocimiento tecnológico y aplicabilidad práctica.

1.2 OBJETIVOS

1.2.1 Objetivo General

Evaluar y repotenciar una plataforma de firewalling de código abierto basado en las características distintivas de un sistema propietario y verificar su comportamiento en términos de performance y efectividad ante ataques informáticos, con el fin reforzar la detección, control y mitigación de ataques de seguridades informáticos en las instituciones gubernamentales.

1.2.2 Objetivos Específicos

- Analizar el marco teórico referencial e identificar las evidencias del estado del

arte.

- Realizar una evaluación cuantitativa de algunos sistemas de firewall de código abierto del performance y efectividad, con respecto a la solución propietaria.
- Repotenciar un sistema de firewalling de código abierto, el de mejor rendimiento y efectividad en la evaluación realizada, cotejando con aquellas funcionalidades distintivas de la plataforma propietaria.
- Evaluar, validar, verificar e interpretar los resultados de efectividad de la solución repotenciada, en una red en producción.
- Publicar los resultados.

1.3 SITUACIÓN ACTUAL DE PLATAFORMAS DE SEGURIDAD - PYMES

De acuerdo con la recomendación RFC 2979 de la IETF (Freed, 2000), se definen las características de comportamiento y requerimientos de interoperabilidad para los firewalls de Internet. Se plantean dichos requerimientos para hacer consistente el comportamiento de los firewalls en distintas plataformas. En dicha recomendación se conceptualiza a un "firewall" como un agente que filtra el tráfico de red de alguna manera, bloqueando el que considera inadecuado, peligroso, o ambas cosas.

Los firewalls son dispositivos de hardware o sistemas de software que controlan el flujo de tráfico entre dos o más redes empleando ciertas políticas de seguridad. Su funcionalidad se limita al cumplimiento de una serie de reglas, con el objetivo de proteger la red interna de las amenazas existentes. Esta tecnología ha evolucionado hasta alcanzar integración a otros sistemas generando nuevos conceptos tales como: Inspección de paquetes *(Deep Packet Inspection, DPI*), que se integra con sistemas de detección y protección de intrusos (IDS/IPS) para analizar el tráfico; Firewalls de la siguiente Generación (*Next Generation Firewall*, NGFW), en los cuales las políticas se

pueden definir basadas en aplicaciones y que combina el NAT, IPS y VPN; y, Gestión Unificada de Amenazas (*Unified Threat Management*, UTM), que consolida varios servicios de seguridad en una sola máquina, tales como VPN, IDS, IPS, Antivirus, Anti- Spam, Web Proxy, NAT, y filtrado de contenido (DragonJAR, 2012).

Existen diferentes formas de proteger una red mediante firewalls: i) Listas de control de Acceso (ACLs), que constituye la forma más sencilla e insegura de topología, en la cual el router desempeña funciones de enrutamiento y filtrado de paquetes; ii) Zona desmilitarizada simple, que puede definirse como un área pública, en la que tanto los servidores públicos como la red interna son protegidos por un firewall; iii) Zona desmilitarizada dual, que adiciona un gateway firewall para controlar y proteger la red interna, una de las razones es la protección de los servidores públicos frente a ataques provenientes de la red interna; iv) Profundidad en defensa, que es un concepto en el que la seguridad se implementa mediante capas para maximizar la protección, así por ejemplo, en el borde del perímetro se ubica un gateway firewall que realiza el filtrado de paquetes y protege la DMZ y LAN de tráfico proveniente de Internet; v) Firewall appliance basado en hardware, que son máquinas optimizadas y diseñadas para realizar trabajos exclusivos de firewall, especialmente de filtrado de paquetes; vi) Firewall basado en software, que constituyen una alternativa más económica en relación a los firewall basados en hardware, pero presentan mayores desafíos en su configuración e implementación, como el IPtables de Linux.

Los criterios para escoger un firewall, de acuerdo con (Newman, 1999) (B. Hickman, 2003) (Yan, Zhang, & Ansari, 2008) (Yongxin, 2011) incluyen: i) Throughput esperado, que es una medida necesaria en la que se pierde en desempeño y se gana en seguridad; ii) Presupuesto, pues existen diversas alternativas libres y

comerciales, dentro de las cuales se les debe evaluar ventajas y desventajas en términos de calidad, soporte, beneficios y costos; iii) Número de redes a proteger, pues el tamaño de la red define las capacidades del hardware. También debe estudiarse y considerarse el tráfico estimado que controlará el firewall; iv) Nivel de profundidad del filtrado: dependiendo de los recursos a proteger es conveniente decidir hasta qué capa debe realizarse un chequeo de los paquetes; y, v) Capacidad de escalabilidad de las funciones del firewall, que es su posibilidad de crecimiento.

Los argumentos anteriormente expuestos son previstos por los fabricantes de soluciones corporativas (Cisco, Checkpoint, Fortinent, Watchward, Nokia, Juniper) en sus diferentes gamas y software de código abierto (Zentyal, Clear OS, Untangle, Pfsense, Smoothwall) que realizan las funciones de firewall (Walter Fuertes P. Z., 2012).

A continuación se realiza una breve descripción de los mecanismos analizados en esta investigación:

Watchward 550-E, es una solución propietaria integral que incluye en la misma caja hardware, implementación de los múltiples servicios y suscripciones para la red perimetral. Su funcionalidad incluye soporte de VPN, SSL, IPSEC, filtrado de contenido, anti-virus, anti-spam, IPS, NAT, capacidades de Networking y gestión Web;

PfSense, basado en sistema operativo FREBSD, es iniciado por Chris Buechler y Ullrich Scott (septiembre de 2004) como una bifurcación (fork) de m0n0wall. De acuerdo a su página oficial (Pfsense, 2012), se ha instalado exitosamente en distintos ambientes, que van desde redes domésticas hasta grandes corporaciones, universidades y otros tipos de organizaciones. Los requerimientos para su instalación son: un ordenador o servidor (CPU - 100 MHz Pentium, RAM - 128 MB, 1 GB disco duro) que

cuente dos tarjetas de red (como mínimo), el proceso de instalación se desarrolla en un ambiente amigable y explicativo, mientras que su gestión es WEB basada en PHP que engloba todas sus funcionalidades, por lo que no es indispensable contar con conocimientos avanzados sobre la línea de comandos UNIX para su manejo.

Zentyal, basado en sistema operativo UBUNTU, empezó como un proyecto colaborativo entre dos empresas y fue publicado como un proyecto de código abierto por primera vez en 2005. El 16 de noviembre de 2006 Zentyal (Zentyal, 2012) (eBoxPlatform) fue oficialmente aprobado como proyecto NEOTEC, recibiendo fondos públicos del CDTI (organización pública española bajo el Ministerio de Industria, Comercio y Turismo) para completar el desarrollo de la versión 1.0.3 Zentyal (eBoxPlatform) fue incluido por primera vez en Ubuntu en 2007, en el GutsyGibbonTribe 3, la tercera versión alfa de Ubuntu 7.10.4 La primera versión candidata a definitiva de Zentyal (eBoxPlatform 1.0), fue publicada en 2008. Esta herramienta es multigestión, puesto que gestiona: la infraestructura de red, como puerta de enlace a Internet (Gateway), gestiona amenazas de seguridad (UTM), gestiona comunicaciones unificadas o una combinación de estas, e incluye un marco de desarrollo (framework) para nuevos servicios en un ambiente WEB.

ClearOS, basado en sistema operativo REDHAT, es una distribución GNU/Linux derivada de ClarkConnect, misma que sintetiza y particulariza aspectos de seguridad derivados del sistema operativo lo que lo hace más atractivo a la hora de definir una distribución especializada en seguridad, dado que tiene las misma capacidades tecnológicas que las herramientas antes mencionadas y soporte de REDHAT que mantiene su desarrollo y performance en continuo crecimiento.

1.4 PLATAFORMAS DE VIRTUALIZACIÓN

La virtualización es una capa abstracta que desacopla el hardware físico del sistema operativo para brindar mayor flexibilidad y utilización de los recursos de TI1. Al separar la operación lógica del hardware físico, un entorno virtualizado proporciona mayor flexibilidad operativa y agiliza los cambios del sistema, ofreciendo una plataforma que refuerza la continuidad del negocio y escala con rapidez para satisfacer las demandas empresariales.

La virtualización permite que múltiples máquinas virtuales con sistemas operativos heterogéneos puedan ejecutarse individualmente, sobre la misma máquina física.

Cada máquina virtual tiene su propio hardware virtual (por ejemplo, RAM, CPU, NIC, disco duro etc.) a través del cual opera el sistema operativo y las aplicaciones.

Las máquinas virtuales al ser un conjunto de archivos, facilitan que se pueda guardar, copiar y proporcionar una máquina virtual de manera rápida. Se pueden mover sistemas enteros (aplicaciones, sistemas operativos, BIOS y hardware virtual completamente configurados) de un servidor a otro.

La virtualización permite implementar recursos informáticos aislando unas capas del sistema de otras: hardware, sistema operativo, aplicaciones, datos, redes, etc. Las tecnologías de la información actualmente han facilitado la evolución de los negocios, pero esto conlleva a mayor complejidad en la gestión de los sistemas. Una de las prioridades de las TI es ayudar a crear infraestructuras queproporcionen flexibilidad y control para proteger los recursos corporativos, cumplir con normas y regulaciones; encontrando el equilibrio perfecto entre ambos requerimientos, es por esto que actualmente se está apostando tan fuerte por la virtualización, que permite crear sistemas más eficientes, flexibles y económicos.

Características principales:

Particionamiento: Ejecuta múltiples máquinas virtuales en un mismo host.

Aislamiento: cada máquina virtual está aislada del resto de máquinas virtuales en el mismo host.

Encapsulación: Las máquinas virtuales encapsulan todo el sistema (configuración de hardware, sistema operativo y aplicaciones) en ficheros.

Independencia del hardware: Una máquina virtual puede funcionar en cualquier servidor, sin modificación.

1.4.1 ENTORNOS VIRTUALES

Una Infraestructura virtual (VI) incluye una nueva capa abstracta entre los servidores (discos, memoria, tarjetas de red, etc.) y programas que están funcionando en estas máquinas.

La infraestructura virtual ordena las operaciones TI permitiendo a las empresas usar y gestionar de forma más óptima los recursos de hardware. Los usuarios ven los recursos como suyos y en cambio los administradores pueden gestionar los recursos a nivel de toda la compañía.

Una máquina virtual representa los recursos físicos de un único computador, mientras que una infraestructura virtual representa los recursos físicos de la totalidad del entorno de TI, agrupando computadores x86, así como su red y almacenamiento asociados, en un pool unificado de recursos de TI.

Estructuralmente, una infraestructura virtual consta de los siguientes componentes:

- Un hipervisor o monitor de máquina virtual (VMM) es una tecnología que está compuesta por una capa de software, que permite utilizar al mismo tiempo

diferentes sistemas operativos o máquinas virtuales en una misma computadora central. Es decir, se encarga de manejar los recursos del sistema principal exportándolos a la máquina virtual.

Hay dos tipos principales:

- Hipervisor nativo. Se ejecuta directamente sobre el hardware y soporta directamente los sistemas operativos paravirtualizados.
- Hipervisor alojado en un SO anfitrión. El software de virtualización se instala sobre un sistema operativo anfitrión.

- Un conjunto de servicios basados en la virtualización que permiten la gestión de recursos disponibles entre las máquinas virtuales alojadas en los servidores.

Soluciones de automatización que proporcionen capacidades especiales para optimizar un proceso de TI como alta disponibilidad, balanceo de carga y un sistema de backup.

1.4.2 MODELOS DE VIRTUALIZACIÓN

La virtualización ha avanzado en los últimos años llegando a impactar soluciones a nivel de usuario (end-to-end) mediante la virtualización de diferentes elementos o componentes computacionales permitiendo simplificar cualquier proceso de administración, un uso más eficiente de los recursos de TI y la flexibilidad para proporcionar los recursos adecuados allí donde se necesiten.

Al momento de implementar una solución de virtualización es importante tener en mente una solución integral apoyada en herramientas de administración como manejo de configuración, de operaciones y la administración dinámica de asignación de recursos, que cubra la mayoría o todos los componentes de virtualización.

Existen varios modelos de virtualización:

- Virtualización de presentación

- Virtualización de aplicación
- Virtualización de servidor
- Virtualización del almacenamiento
- Virtualización de la red
- Virtualización de la estación de trabajo
- Paravirtualización

1.4.3 SELECCIÓN DE MODELO - VIRTUALIZACIÓN DE ESTACIÓN DE TRABAJO

Este modelo se ha seleccionado por su adaptabilidad, usabilidad y parametrización de aspectos técnicos necesarios para el desarrollo del proyecto, debido a su funcionamiento, basado en software, que permite crear equipos virtuales mismos que emulan los servicios y capacidades del hardware subyacente, permitiendo a un usuario correr más de un sistema operativo, con diferentes versiones, en la misma estación de trabajo, dando a cada instancia de sistema operativo un ambiente aislado, usando los recursos de máquina física (Figura 1).

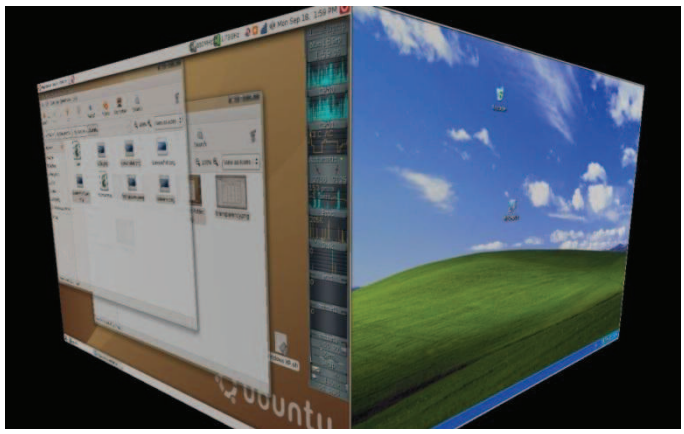

Figura 1. Virtualización estación de trabajo Ubuntu-VirtualBox

Fuente: http://aulaweb.uca.edu.ni/blogs/cleal/files/2010/07/vmware1.jpg

El sistema operativo anfitrión administra el hardware y actúa como interfaz con los sistemas operativos huéspedes o invitados. El usuario puede navegar desde una máquina virtual con su sistema operativo hacia otra, como lo haría entre ventanas de diferentes aplicaciones dentro de un mismo sistema operativo.

La virtualización en esencia es una técnica para compartir recursos de hardware. Puede ser utilizado para equipos de partición física para soportar múltiples máquinas virtuales, su interconexión, y compartir recursos de hardware, como dispositivos de CPU, memoria y entrada / salida. Se proporciona una capa de abstracción adicional entre el hardware y sistema operativo (OS). La técnica permite, a través de hardware, de tener varios sistemas operativos invitados de diversos tipos de ejecución al mismo tiempo. En la actualidad, existen varias alternativas de software que hacen posible la virtualización, una de estas herramientas es Virtual Box, desarrollada por un equipo de investigadores y supervisado por ORACLE. Virtual Box es un software de virtualización X86 que se desplegarán en las máquinas virtuales destinados a los ordenadores de sobremesa y servidores de la empresa. Virtual Box permite la ejecución de sistemas operativos sin modificaciones, incluyendo todo el software instalado en ellos]. En este trabajo, Virtual Box proporciona la infraestructura para implementar y administrar un VNE que puede ser configurado para emular la ejecución de ataques de escaneo de puertos.

1.5 FIREWALLS

Los Firewalls o cortafuegos son considerados como uno de los elementos proactivos y esenciales de la seguridad de redes y sistemas de información, los mismos han sido utilizados ampliamente en la defensa de tráfico sospechoso y en el control del permitido. Posicionado en la frontera de una red privada e Internet, un firewall examina

todos los paquetes entrantes y salientes en función de las normas de seguridad definidas por personal especializados en base a un conjunto de reglas o políticas previamente analizadas y aprobadas para el filtrado, ya sea este a nivel de protocolo o de aplicación, en la red (Walter Fuertes P. Z., 2011). En principio provee dos servicios básicos: bloqueo y acceso de tráfico.

Los Sistemas de Información han tenido una continua evolución en los campos de conectividad y seguridad, este segundo aspecto es considerado fundamental dentro de las organizaciones y su información, planeando así estrategias de seguridad para minimizar las vulnerabilidades que en ella puedan presentarse (Figura 2)

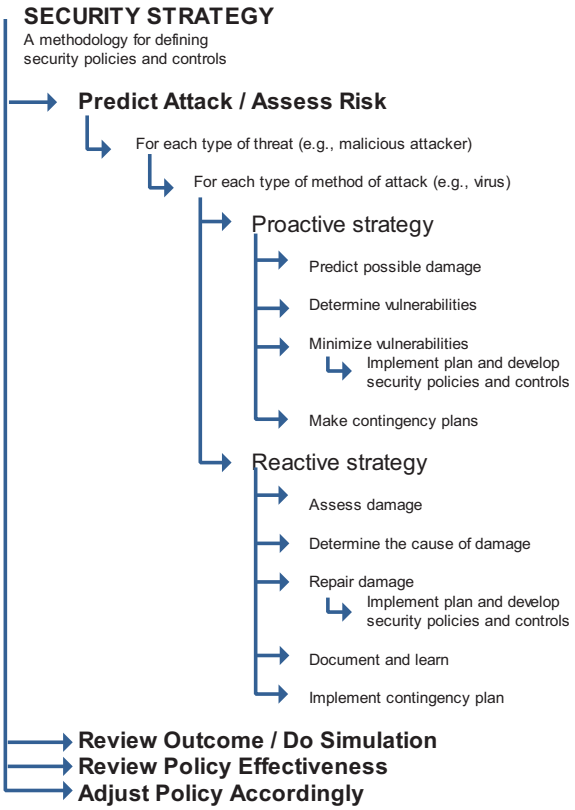

Figura2. Estrategias de seguridad

Fuente: http://technet.microsoft.com/en-us/library/cc723503.aspx

CAPITULO II

PRINCIPIOS DE DISEÑO DE FIREWALLS E IMPLEMENTACIÓN DE INTERFACES WEB PARA SU ADMINISTRACIÓN

Los Firewalls están insertados entre la red local y la Internet (red no confiable), y su funciones son: establecimiento de enlaces controlados, protección de ataques internos y externos a la RED y ser un único punto de convergencia (Figura 3).

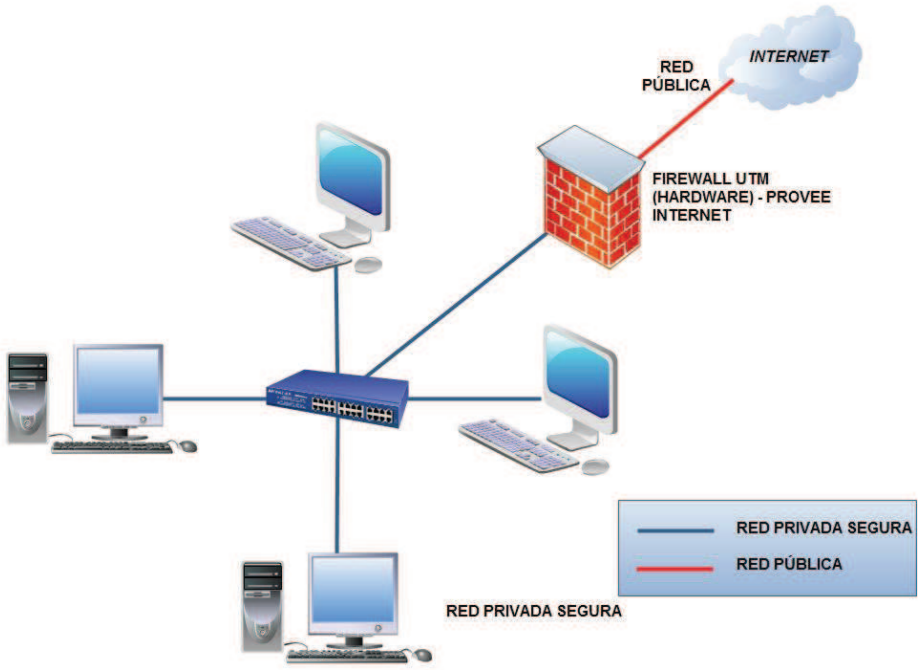

*Figura 3.*Diagrama –único punto de convergencia

2.1 ESTABLECIMIENTO DE ENLACES CONTROLADOS

Todo el tráfico interno hacia el exterior debe pasar a través del firewall, sólo el tráfico autorizado (definido dentro de una política de seguridad local) será permitido pasar, a través de filtros y gateways.

Protección de ataques internos y externos a la RED.- El firewall es mentalizado como un equipamiento inmune a penetraciones, robusteciendo cada vez más sus sistemas operativos sean estos propietarios o de código abierto.

2.1.1 Único punto de convergencia.

Al ser un equipamiento de frontera y único punto de choque el nivel de monitorización, redundancia y control deben ser priorizados en el diseño. Para esto existen políticas por defecto: se permite todo el tráfico excepto aquel explícitamente negado o se niega todo lo que explícitamente no se permita (Stallings, 2009). Evidentemente la segunda política es mejor en cuanto a seguridad pero a la hora de trabajar los usuarios en muchas ocasiones están limitados. En base a las reglas implementadas se determina la eficiencia o no del equipamiento, considerando un número limitado de reglas que en lo posible cubran la política escogida, caso contrario perderá demasiado tiempo en evaluar los paquetes. La gestión natural de reglas y tolerancia a errores de gestión es también muy importante sobre todo cuando la red es de un tamaño considerable.

Al hablar de redundancia y control es importante considerar el aspecto económico, que conlleva cubrir estos aspectos determinando si la información que converge en la RED es realmente delicada para la organización o no y la granularidad máxima, que será ajustada lo más posible al nivel de detalle exigido por las políticas de la organización (Hongxin Hu, 2012).

2.1.2 Técnicas generales para Control Accesos

Los firewalls en la actualidad son intuitivos y amigables en su configuración, sin embargo se debe hacer hincapié en los siguientes aspectos básicos:

- Control de Servicios: determina tipo servicios de Internet que pueden ser

accedidos.

- Control de Dirección: determina la dirección en que se permiten fluir requerimientos de servicios particulares.
- Control de Usuarios: controla acceso a servicio acorde a permisos de usuarios.
- Control de Comportamiento: controla el uso de servicios particulares, mail, http, etc.

2.1.3 TIPOS DE FIREWALLS

Los firewalls son clasificados como (Christoph L. Schuba. On the Modeling, Design, and Implementation of Firewall Technology.Doctoral dissertation, PurdueUniversity, December 1997):

- Router con Filtrado de Paquetes
- Gateway a Nivel de Aplicación
- Gateway a Nivel de Circuitos

2.1.3.1 Router con Filtrado de paquetes

Este tipo de modalidad en aspectos de seguridad, es desarrollado con la aplicación de reglas para cada paquete entrante y luego lo reenvía ó descarta. Este filtro es bidireccional y seteado típicamente como una lista de reglas (ACL: Access Control List) basadas en "matches" de campos de cabeceras IP ó TCP/UDP, por defecto: niega (discard/deny) ó envía (forward/allow), para lo cual es recomendable habilitarla explícitamente como no permitida (default= deny) [Anexpertsystemforanalyzing firewall rules PasiEronen and JukkaZitting Helsinki University of Technology {pasi.eronen, jukka.zitting}@hut.fi].

Se puede resumir como parte de las ventajas de este modelo: su simplicidad, transparencia hacia usuarios, alta velocidad, menor coste de implementación, es más

fácil denegar y permitir solo unos cuantos servicios que permitirlo todo y luego denegar todo aquello que pueda ser una amenaza para la seguridad, existe la posibilidad de rechazar, que es como denegar pero se envía un paquete ICMP para avisar que este paquete ha sido rechazado, en consecuencia esto supone una sobrecarga de la red y una información muy útil para los hackers y como desventajas se presenta mucha dificultad a la hora de setear reglas de filtrado (configuración) y un pobre manejo de Autenticación

2.1.3.2 Gateway a Nivel de Aplicación

Suele ser un ordenador especial que examina las direcciones de los paquetes para determinar si el paquete debe pasar a la red local o no. El firewall utiliza la información contenida en la cabecera del paquete para controlar el acceso del mismo. La configuración de un firewall puede parametrizarce para el bloqueo de todos los mensajes que provengan de un sitio determinado, así como todos los paquetes destinados a una dirección. Para el funcionamiento del mismo, se debe configurar al firewall el bloqueo de dichos paquetes con información que contengan la dirección de los sitios (de destino). Con este sistema se puede bloquear todo un sitio, es decir, toda una red entera. Y con configuraciones adicionales se podrá reconocer y ejecutar las acciones detalladas en él, como manejo de usuarios para el acceso a Internet con restricción de FTP e incluso para que pueda descargar archivos sin posibilidad de transferir archivos al servidor. De una manera básica la configuración para que la comunicación sea transparente para el usuario final deberá contener:

- Dirección de origen/destino de los datos.
- Protocolo de sesión de los datos. TCP, UDP o ICMP.
- Si el paquete es el inicio de una petición de conexión.

- El puerto de aplicación de origen/destino del servicio deseado.

2.1.3.3 Servidores proxy o firewalls a nivel de aplicación

Su principal función es controlar el tráfico entre dos redes, es decir, se comunican con servidores exteriores a la red, y la intranet no mantiene conexión alguna con Internet. En lugar de conectarse, el tráfico que fluye de una red a otra nunca interactúa con el tráfico de una red adicional. El servidor de una red transmite una copia aislada de cada paquete autorizado. Estos firewalls enmascaran el origen de la conexión inicial y protegen la red frente a los usuarios de Internet que intentan recopilar información de su red privada. Una ventaja destacable de un firewall es poseer la capacidad de reconocimiento de protocolos de red, y el mismo parametrizarce para el control de los mismos. Como desventajas, se debe contar con software adicional en cada cliente para que pueda interactuar con el proxy y la degradación del performance de la RED, puesto que, es un programa (Figura 4) el que analiza la validez o no de los paquetes transmitidos.

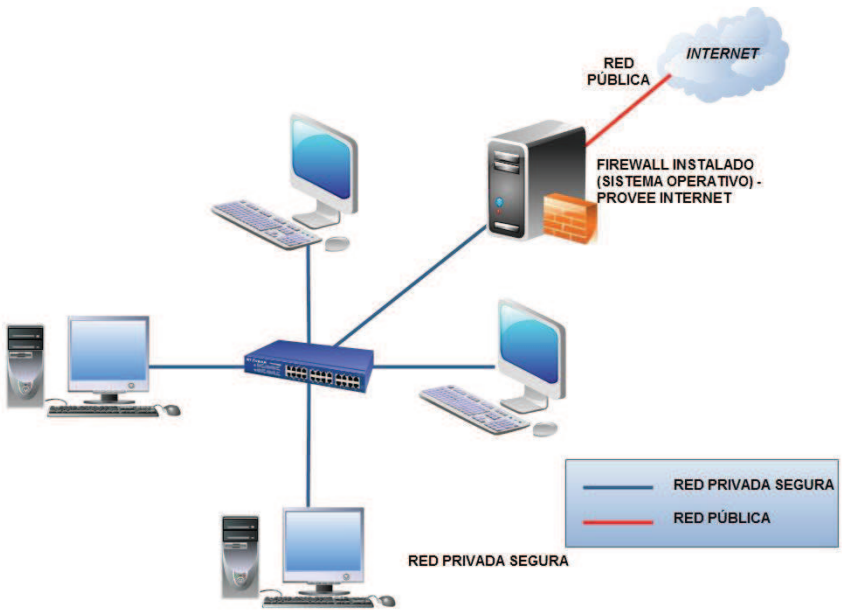

*Figura 4.*Red configurada con Firewall

2.1.3.4 Firewalls a nivel de circuito

Este tipo es similar a los firewalls a nivel de aplicación, puesto que ambos son proxies y se debe tener en cuenta que los firewalls a nivel de aplicación requieren la utilización de software de proxy especial para cada servicio que se desee incluir en la red, como FTP o HTTP (Schuba, 1997). Por el contrario este tipo de firewalls crea un circuito entre el cliente y el servidor sin necesidad de que la aplicación sepa nada del servicio. Con este método se protege el inicio de la transacción sin interferir en la transacción que se esté realizando. La principal ventaja reside en que proporciona servicios para una gran variedad de protocolos.

2.1.4 ACTUALIDAD DE FIREWALLS PARA PYMES

El acceso de las PYMES a nuevas tendencias tecnológicas de seguridad cada vez son más costosas pero obligatorias e imperativas ante amenazas que cambian y evolucionan constantemente en busca de la vulneración de las seguridades, desafiando a los administradores de RED a la búsqueda de mejores herramientas que condensen sus requerimientos técnicos de comunicaciones (comunicaciones unificadas) en un único equipo o appliance, de fácil configuración(WEB), continua reportería de incidentes al operador (vía mail), soporte y actualizaciones de seguridad.

Bajo estos aspectos se definen dos tipos de FIREWALL:

- Gestión unificada de amenazas – appliance(UTM)
- Plataformas OPENSOURCE de seguridad

2.1.5 GESTIÓN UNIFICADA DE AMENAZAS - APPLIANCE (UTM)

Los dispositivos integrados de seguridad UTM (Appliance), son considerados como elementos robustos integrales, puesto que conjugan el hardware de conexión como un sólido sistema operativo sobre el cual se colocan diversos aspectos de

seguridad. La estructura y características dependen de cada fabricante, y también puede variar en cada modelo, pero entre las más comunes tenemos el firewall o la protección contra intrusiones y otras consideradas opcionales como antivirus, anti-spam, VPN. Es decir, lo necesario para contar con una buena línea de defensa perimetral de la empresa, en la Figura 5 se presenta una configuración típica de dispositivo UTM con algunas de las utilidades antes mencionadas.

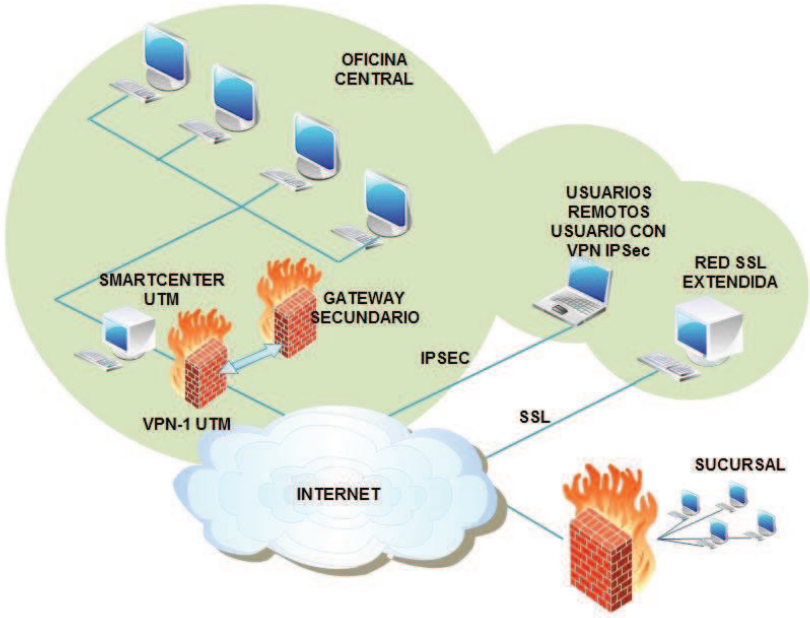

*Figura 5.*Configuración de dispositivo UTM

El origen del término UTM (Unified Threat Management) se le otorga a Charles Kolodgy de la IDC (Internacional Data Corporation) en el año 2004, dicho término es usado para describir firewalls capaces de desempeñar múltiples funciones adicionales que los dispositivos tradicionales no, entre ellas se pueden mencionar filtro de contenido web, antivirus, anti-spam, IDS e IPS. Todo lo anterior está relacionado íntimamente con la función de proxies que estos dispositivos utilizan, analizando y enrutando todo el tráfico de la red, aunque de igual manera estos pueden funcionar de modo transparente

(es decir no realizar ruteo) sin utilizar totalmente a plenitud sus capacidades Los sistemas de gestión unificadas de amenazas (UTM), se han convertido en un importante elemento para garantizar la protección perimetral de cualquier empresa por pequeña que esta sea.

En lugar de aplicar varios elementos diferentes para lograr las diversas capas de protección para la parte de conexiones al exterior, resulta más eficaz, compacto, y con ello barato, colocar un elemento que agrupe la mayoría, o todas las funciones necesarias o al menos a las más importantes.

El empleo de un dispositivo de seguridad ofrece la indudable ventaja de usar un hardware idóneo, normalmente reducido al mínimo, y colocar el software específico destinado a cubrir diversos aspectos de seguridad. En lugar de emplear un PC, o un servidor, convencional y montar sobre el mismo un sistema operativo estándar y luego diversas aplicaciones de seguridad, un dispositivo parte de un circuitería reducida, en la que se excluye todo elementos que puedan causar fallos o dejar abiertas puertas no deseadas.

La centralización de los elementos de seguridad en un solo dispositivo presenta grandes ventajas, aunque también algún inconveniente. Por ejemplo, se concentra toda la seguridad y conectividad en un solo elemento, con el inconveniente que si este falla, todo el sistema de seguridad se cae, e incluso todas las conexiones quedan interrumpidas. Para paliar este efecto hay diversas soluciones, como la colocación de una pareja de elementos, uno activo y otro en reserva o en balanceo de carga entre ambos que permite que el dispositivo que funciona absorba directamente la carga del que falla o está siendo sustituido. Lo anterior se conoce como HA (*High Availability*).

La gestión de estos equipos es un punto importante. Cuanto más centralizada y simple, más rápido, económico y fiable es crear una adecuada defensa. Además de contar con una combinación de hardware y software especializada, con sus cualidades de potente y fiable, es obligatorio que la gestión sea sencilla. La composición de los diversos elementos de seguridad que se colocan sobre un determinado dispositivo es notablemente amplia y varía mucho para cada fabricante e incluso dependiendo de gamas o modelos. El elemento más básico es un cortafuegos o firewall, que es el elemento primordial para establecer una defensa perimetral de red. Luego, capa sobre capa, se colocan otras funciones, como antivirus, anti-spam, detección de vulnerabilidades y filtrado selectivo de contenidos Web.

Una de las grandes ventajas del uso de un dispositivo es que se libera a los host finales de gran parte de la carga de trabajo en las funciones de seguridad. Al menos para todos los que están conectados dentro de la oficina. Un dispositivo de esta clase no elimina la necesidad de contar con protección en cada puesto de trabajo, pero descarga a éstos de gran parte de su trabajo.

2.1.5.1 Cuadrantes Mágicos de Bussines Intelligence Empresarial

Un cuadrante mágico es una herramienta analítica creada y promovida por la empresa Gartner, el cual muestra una representación gráfica del mercado compartido en un determinado periodo de tiempo. Los Cuadrantes Mágicos de Gartner proporcionan a las empresas un medio para identificar y diferenciar a los proveedores de servicios del sector de las tecnologías de la información.

Según define Gartner, los líderes en los cuadrantes mágicos son aquellos fabricantes de tecnología que operan bien hoy en día, tienen una visión clara de la

dirección del mercado y desarrollan activamente las competencias necesarias para mantener su posición de líderes en el mercado.

A continuación (Figura 6) se presenta un cuadrante mágico proporcionado por Gartner con fecha de noviembre de 2004. Este hace referencia a las plataformas de Business Intelligence.

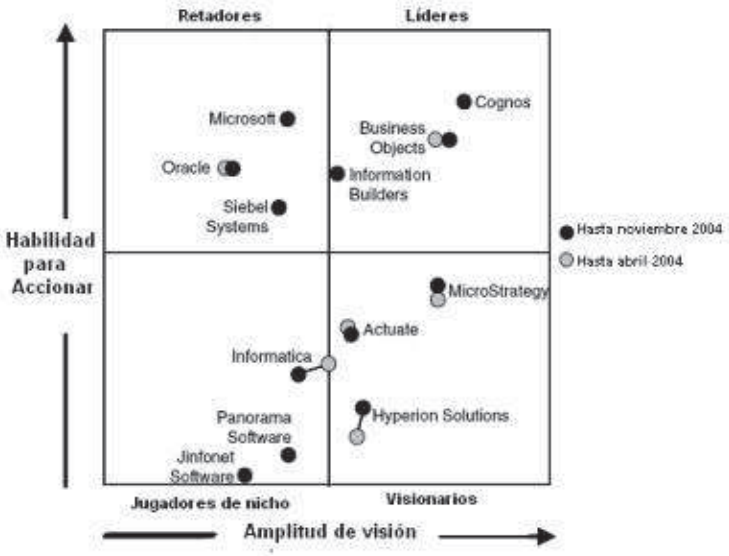

Figura 6 Cuadrante mágico de las plataformas de BI

Fuente: Gartner Research (Noviembre 2004)

La manera de interpretarlo según especialistas determina que aquellos que figuran en el cuadrante principal pueden ofrecer un gran servicio prácticamente a cualquier cliente. Otros podrían ser empresas que abasteciesen nichos de mercado, por ello las notas tratan sobre los nichos de cada una de las empresas y describen los 'puntos favorables' de todas ellas.

En este caso, se está hablando principalmente de grandes clientes corporativos. Un Cuadrante Mágico no deja de ser potencialmente útil para pequeñas y medianas empresas (PYMES), pero éstas posiblemente tengan que calibrar aspectos adicionales

como, por ejemplo, 'el modo en que se dicho proveedor concreto se pondría en contacto conmigo'

El cuadrante mágico debe tomarse como una herramienta y no como una guía específica de acción. En el caso de Business Intelligence Empresarial, el gran visionario hasta noviembre de 2004 es COGNOS. (Gartner, 2004).

2.1.5.2 Watchward

Como antecedente es importante mencionar que la disponibilidad física de este UTM determinó la selección de dicha herramienta en la elaboración del proyecto y en base a sus capacidades se determinó las directrices básicas de seguridad que debían cumplir los firewall OPEN SOURCE en funcionalidad y performance para el desarrollo de la investigación, cabe mencionar que WATCHGUARD es una solución líder (cuadrante de Gardner) en aspectos de seguridad, estabilidad, soporte técnico, entre otros, convirtiéndose así la presente investigación en un reto ambicioso y técnicamente alcanzable.

Esta solución según el cuadrante mágico de Gartner del 2012 (Figura 8), al igual que Fortinet y SonicWall, se encuentra como unos de los proveedores vanguardistas en aspectos de fabricación y venta de sus productos de firewall multifunción para PYMES al ofrecer nuevas características de seguridad a la hora de explotar sus vulnerabilidades, demostrando así fiabilidad, rendimiento, consistencia, fácil gestión y administración, a un bajo coste sin comprometer el rendimiento de la red.

Figura 7. UTM Watchguard

Fuente: http://www.watchguard.com – Firebox Edge

Entre los aspectos más relevantes a ser tomados en cuenta para esta categorización se encuentran:

- Amplia gama de modelos que cubran las necesidades de las pymes.
- Soporte de múltiples características en seguridad.
- Capacidad de gestión, y;
- Reportes de fácil manejo y parametrización.

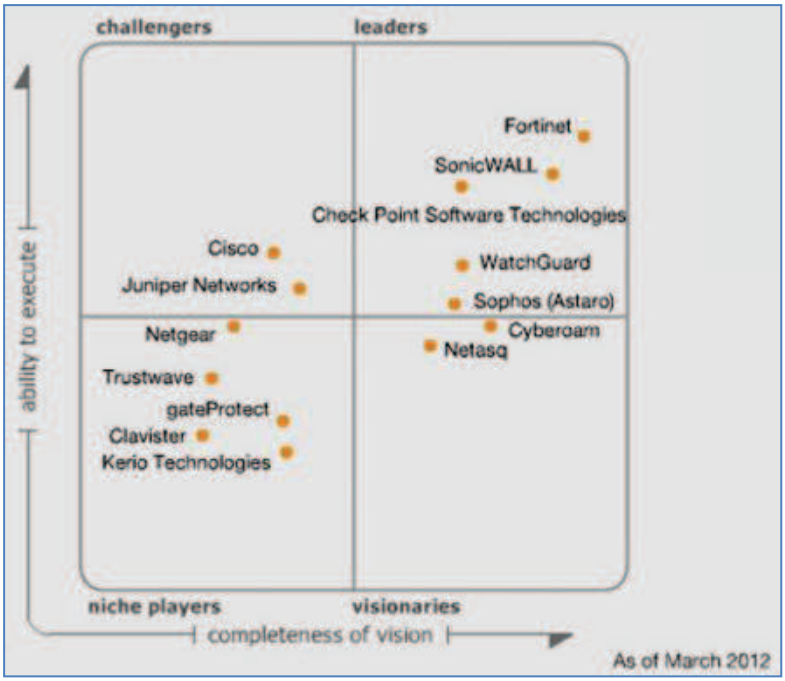

*Figura 8.*Cuadrante de Gartner
Fuente: Gartner 2012

2.1.5.3 Características Watchward 550-E

Funcionalidad de Firewall.

- Soporte de tres zonas de seguridad: externa, privada y DMZ.
- Soporte de direcciones IP estáticas y dinámicas.
- Throughput de Firewall 100 Mbps
- 10000 sesiones concurrentes.
- Implementación de políticas de seguridad capa de aplicación (capa 7) (proxy transparente).
- Políticas de seguridad en capa de aplicación pre-configuradas para protocolos comunes: http, https, POP3, SMTP, FTP, TFTP, DNS, SIPv2, H323.
- Creación de perfiles de usuario.
- Soporte de servicio Dynamic DNS.

Soporte de VPN.

- Soporte de VPNs Móviles (Usuario – Equipo).
- Soporte de 10 VPNs Móviles usando protocolo IPSec.
- Soporte de VPNs entre oficinas (Equipo – Equipo).
- Soporte de 10 VPNs entre oficinas usando protocolo IPSec.
- Mecanismos de autenticación soportados: DES, 3DES, AES 128-, 192-, 256-bit
- Mecanismos de encriptación soportados: SHA-1, MD5, IKE Pre-Shared Key.

Filtraje de Contenido

- Filtraje de contenido categorizado
- Suscripción anual de antivirus, filtrado de contenidos y antispam.
- Establecimiento de excepciones para el filtrado de contenido.

Anti-virus

- Soporte de Anti-virus.
- Actualización del Antivirus automatic
- Soporte de bloqueo de Spyware.
- Escaneo de archivos comprimidos (.zip, .tar)
- Soporte para los principales protocolos: HTTP, FTP, SMTP, and POP3.

Anti-spam

- Soporte Anti-spam.
- El Antispam debe tener servicios de cuarentena.
- Capacidad de bloquear spam basado en imágenes además de spam basado en texto

IPS

- Soporte IPS en el mismo equipo.
- La actualización del IPS automática.
- Análisis en capa de aplicación definiendo el nivel de severidad del ataque.
- Bloqueo automático de fuentes conocidas de ataques.
- Soporte para los principales protocolos: HTTP, FTP, SMTP, and POP3

NAT

- Soporte NAT.
- Soporte de NAT estático (Port Forwarding)
- Soporte de NAT dinámico.
- Soporte de NAT basado en políticas.

Capacidades de Networking

- Número de interfaces: 4 interfaces 10/100 BaseT.
- Soporte de VLANs: 15 VLAN.

- Control de ancho de banda (upload) por usuarios, políticas, protocolo, grupo de usuarios.
- Utilización del ancho de banda utilizado por interfaces.
- Soporte en modo de enrutador (routing) y en modo drop-in (transparente o bridge).
- En modalidad Router/NAT deberá tener soporte para Static Route, Dynamic Route (RIP, OSPF y BGP4), NAT 1-to-1.
- Soporte para enrutamiento basado en políticas.

Gestión

- Interfaz de administración gráfica en tiempo real.
- Edición de políticas de seguridad fuera de línea.
- Soporte de tres roles para operadores: Administrador, Monitoreo y Configuración, Sólo Monitoreo.

Logs y reports

- Almacenamiento externo de Logs.
- Base de datos SQL.
- Logs encriptados

2.1.6 PLATAFORMAS OPENSOURCE DE SEGURIDAD

Las PYMES, luego de un análisis costo/beneficio optan por asegurar sus redes con distribuciones especializadas en seguridad, bajo sistemas operativos reconocidos: REDHAT, UBUNTU, BSD, entre otros. Dichas soluciones ofrecen entre sus beneficios:

- Funcionalidades de Firewall.
- Soporte de VPN.

- Filtraje de Contenido
- Anti-virus
- Anti-spam
- IDS/IPS
- NAT
- Capacidades de Networking - VLANS
- Gestión
- Logs y reportes

Cabe destacar que estas distribuciones demandan bajos requerimientos de hardware dependiendo el throughput deseado y se diferencian por el tipo de sistema de archivos de disco.

La Tabla 1 muestra los requerimientos de hardware que Zentyal necesita tanto de procesamiento, memoria, disco e interfaz de red dependiendo de la cantidad de usuarios que accedan a él.

Perfil de Zentyal	Usuarios	CPU	Memoria	Disco	Tarjetas de red
Puerta de acceso	<100	P4 o equivalente	2G	80G	2 ó más
	100 ó más	Xeon Dual core o equivalente	4G	160G	2 ó más
UTM	<100	P4 o equivalente	1G	80G	1
	100 ó más	Xeon Dual core o equivalente	2G	160G	1
Infraestructura	<100	P4 o equivalente	1G	80G	1
	100 ó más	P4 o equivalente	1G	160G	1
Oficina	<100	P4 o equivalente	1G	2100G	1
	100 ó más	Xeon Dual core o equivalente	2G	1000G	1
Comunicaciones	<100	Xeon Dual core o equivalente	4G	2100G	1
	100 ó más	Xeon Dual core o equivalente	8G	1000G	1

Tabla 1. Demanda de Hardware – ZENTYAL

Fuente: http://doc.zentyal.org/ - REQUERIMIENTOS DE HARDWARE

La Figura 9 presenta los requerimientos de hardware que ClearOs demanda de procesamiento, memoria, y disco, dependiendo de la cantidad de usuarios concurrentes.

RAM and CPU	5 users	5-25 users	25-50 users	50-250 users	250+ users
Processor/CPU	Low-Power	Basic	Dual-Core	Quad-Core	Multi-Core + Multi-Processor
Memory/RAM	1 GB	2 GB	4 GB	8 GB	16-32 GB
Hard Disk					
Hard Disk	Installation and logs require 1 GB - optional storage is up to you				
RAID	Recommended for mission critical systems				

Figura 9. Demanda hardware – CLEAROS

Fuente: http://www.clearcenter.com/support - HARDWARE REQUIREMENTS

La Figura 10 resume el dimensionamiento de hadrware en relación a las consideraciones del rendimiento requerido y las características que serán usadas por esta distribución (Pfsense).

Hardware Sizing Guidance

When sizing hardware for use with pfSense, two main factors need to be considered.

- Throughput required
- Features that will be used

Throughput Considerations

If you require less than 10 Mbps of throughput, you can get by with the minimum requirements. For higher throughput requirements we recommend following these guidelines, based on our extensive testing and deployment experience. These guidelines offer a bit of breathing room because you never want to run your hardware to its full capacity.

10–20 Mbps – No less than 266 MHz CPU
21–50 Mbps – No less than 500 MHz CPU
51–200 Mbps – No less than 1.0 GHz CPU
201–500 Mbps – server class hardware with PCI-X or PCI-e network adapters, or newer desktop hardware with PCI-e network adapters. No less than 2.0 GHz CPU.
501+ Mbps – server class hardware with PCI-X or PCI-e network adapters. No less than 3.0 GHz CPU.

Feature Considerations

Most features do not factor into hardware sizing, though a few have significant impact on hardware utilization.

VPN – Heavy use of any of the VPN services included in pfSense will increase CPU requirements. Encrypting and decrypting traffic is CPU intensive. The number of connections is much less of a concern than the throughput required. A 266 MHz CPU will max out at around 4 Mbps of IPsec throughput, a 500 MHz CPU can push 10–15 Mbps of IPsec, and relatively new server hardware (Xeon 800 FSB and newer) deployments are pushing over 100 Mbps with plenty of capacity to spare. Supported encryption cards, such as several from Hifn, are capable of significantly reducing CPU requirements.

Captive portal – While the primary concern is typically throughput, environments with hundreds of simultaneous captive portal users (of which there are many) will require slightly more CPU power than recommended above.

Large state tables – State table entries require about 1 KB of RAM each. The default state table, when full at 10,000 entries, takes up a little less than 10 MB RAM. For large environments requiring state tables with hundreds of thousands of connections, ensure adequate RAM is available.

Packages – Some of the packages increase RAM requirements significantly. Snort and ntop are two that should not be installed on a system with less than 512 MB RAM.

Figura 10. Demanda hardware – PFSENSE

Fuente: http://www.pfsense.org/ - HARDWARE SIZING GUIDANCE

2.1.6.1 SISTEMA DE ARCHIVOS DE DISCO

Para la presente investigación es muy importante conceptualizar los sistemas de archivo de disco que utiliza cada una de las distribuciones, puesto que en base a su diseño los sistemas mejoran su performance en escritura, almacenamiento, seguridad y recuperación de la información. Existen múltiples sistema de archivos en UNIX, cada distribución incluye el soporte de algunos y por defecto formatea el disco duro con el más estable, considerado el más óptimo en rendimiento:

a. EXT4 (ZENTYAL - CLEAROS)

Ext4 añade varias funcionalidades al sistema de particionamiento Ext3. Gracias al direccionamiento de 48 bits, el tamaño máximo del sistema de archivos aumenta a un megabyte (alrededor de mil millones de gigabytes), y el tamaño máximo de cada archivo es ahora de 16 terabytes (alrededor de 16000 gigabytes); sin embargo, en la práctica, el sistema de archivos solo puede tener actualmente hasta 16 terabytes de tamaño. Otro importante cambio introducido con el Ext4 es el uso de extents, que consisten en bloques adyacentes físicamente, que son marcados como tales por el sistema y tratados como un solo bloque, ahorrando inodes así como las operaciones necesarias para manejarlos. Además se ha optimizado la asignación de bloques, permitiendo asignar múltiples bloques en una sola instrucción, retrasar la asignación hasta que la escritura del archivo se haga efectiva, o manejar la pre-asignación de bloques que algunos programas requieren en forma más eficiente. El journal también se ha optimizado, utilizando sumas de verificación para saltarse diversos pasos necesarios en el Ext3. Incluso se puede desactivar el journal del todo para aumentar ligeramente la velocidad [Matthew Blizek y otros Ext4. Linux KernelNewbies. Disponible en http: //kernelnewbies.org/Ext4 Consultado el 19 de setiembre de 2010.].

b. UFS (por defecto PFSENSE)

Unix File System (UFS) es un sistema de archivos utilizado por varios sistemas operativos UNIX y POSIX. Es un derivado del Berkeley Fast File System (FFS), el cual es desarrollado desde FS UNIX (este último desarrollado en los Laboratorios Bell). BSD optimizó esto en el FFS invirtiendo los grupos de cilindros, dividiendo el disco en grupos más pequeños, cada uno con su propio grupo de inodos y bloques de datos.

La mayoría de sistemas derivados de BSD, incluyendo a FreeBSD sistema base de PFSENSE, utilizan una variante de UFS.

2.1.7 APLICACIONES DE SEGURIDAD OPEN SOURCE

2.1.7.1 INTRODUCCIÓN

Para la repotenciación de funcionalidades, en aspectos de seguridad del firewall OPEN SOURCE, es necesario recurrir a desarrollos, bajo Linux, puesto que los mismos han evolucionado ampliamente en aspectos de seguridad debido a su amplia escalabilidad, los enfoques de cada uno de estos desarrollos van desde análisis exhaustivos de paquetes, escáneres de vulnerabilidades, obtención de información inalambrica de seguridad, entre otros. Estos paquetes al instalarse y configurarse adecuadamente pueden repotenciar un sistema base de seguridad:

Snort, Kismet, ntop, Nagios, EmergingThreats, JasperReports, Nmap, OSSEC, OSVDB, NFSen/NFdump, OCS, Pads, P0f, Arpwatch, Nepenthes, psad, Tcpdump, nessus, entre otros. A continuación se destacará los utilizados en el proyecto:

2.1.7.2 SNORT

Los sistemas Linux especializados en seguridad (CLEAROS – ZENTYAL - PFSENSE) incluyen Snort, mecanismo de alerta IDS, para el análisis de paquetes

(sniffer) y detección de intrusos basado en red (se monitoriza todo un dominio de colisión). Catalogado como un software flexible, mismo que ofrece capacidades de almacenamiento de sus bitácoras tanto en archivos de texto como en bases de datos abiertas como lo es MySQL. Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida.

En funcionamiento como IDS implementa un lenguaje de creación de reglas flexibles ((Zitting, 2001)), potentes y sencillas. Durante su instalación ya nos provee de cientos de filtros o reglas para backdoor, DDoS, finger, FTP, ataques web, CGI, Nmap.

2.1.7.3 P0f

Como metodología preventiva, los sistemas de seguridad no solo deben estar dotados de sistemas que tengan funcionalidades de mitigación de ataques de manera reactiva, sino también de sistemas que permitan analizar potenciales equipos maliciosos conectados a la RED obteniendo información clave como el sistema operativo y versionamiento del mismo, P0f cumple con este objetivo al realizar identificación pasiva de sistema operativo, permite detectar el sistema y la versión de las maquinas conectadas a la Red. P0f también realiza tareas de medición física aproximada de un sistema remoto, detección de balanceadores de carga, distancia de sistemas remotos y tiempo de funcionamiento, detección de firewall e Identificación del tipo de conexión de un equipo remoto (DSL, OC3, aviancarriers) y su ISP (Figura 11).

Un aspecto muy importante es que no genera ningún tráfico en la red, gracias a esto es posible identificar el sistema de equipos que estén detrás de un cortafuego donde un escanner habitual no llegaría, es catalogado como un sistema liviano, rápido.

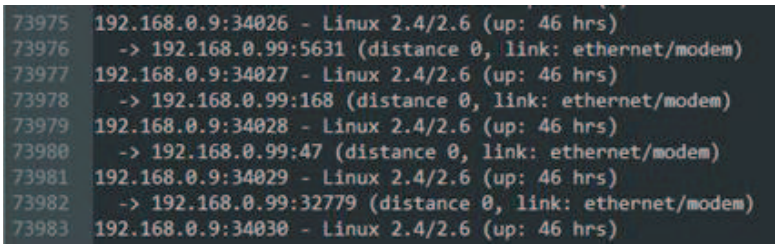

*Figura 11.*P0f

Fuente: http://lcamtuf.coredump.cx/p0f3/ - p0f

2.1.7.4 TCPDUMP

La captura del tráfico bajo demanda en una RED, permite analizar potenciales ataques o tráfico mal intencionado, sin que esto afecte el performance del Hardware del Firewall, Tcpdump es una herramienta en línea de comandos que cumple con este objetivo (Figura 12). Esta herramienta es altamente utilizada para la depuración de aplicaciones que utilizan la red y en algunos casos para la captura de datos no cifrados enviados a través de la RED, puesto que, protocolos como telnet y HTTP no cifran los datos que por ellos convergen, y en algunos casos se obtienen contraseñas dando un mal uso a la herramienta.

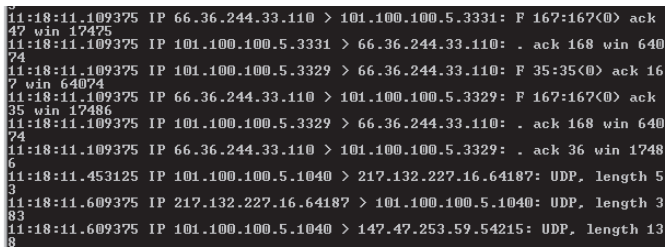

Figura 12. TCPDUMP

Fuente: Imagen capturada en el desarrollo proyecto - TCPDUMP

2.1.7.5 PSAD

Los IDS basados en SNORT, simplemente detectan o alertan actividades consideradas maliciosas o dañinas para la RED, es en este ámbito donde se requiere vincular la detección con el bloqueo. PSAD (Port ScanAttack Detector) realiza esta

función mejorando así la seguridad reactiva de los sistemas firewall. PSAD no trabaja únicamente con SNORT en su campaña de securizar la RED, sino que también interopera con iptables para detectar los portscans y otros tráficos sospechosos (Psad, 2012). Al operar con IPTABLES, una vez configurado, PSAD registra toda actividad que por ella converja, de esta manera PSAD detecta y bloquea los portscan u otros tráficos sospechosos. Para TCP scanspsad analiza las banderas TCP para determinar el tipo de scan (syn, fin, xmas, etc.) y para determinar las opciones de línea de comando correspondientes que pudieran usarse para que nmap genere tal scan (Figura 13).

Esta herramienta hace uso de varias firmas TCP, UDP e ICMP contenidas en el sistema de detección de intrusos Snort para detectar tráfico sospechoso tales como backdoors, herramientas DDoS, identificación de OS, entre otras.

PSAD interopera su actividad con alertas generadas por Snort (Beale, 2004), mismas que son detectadas y directamente bloqueadas por iptables a través del uso de un juego de reglas generadas por fwsnort (Xie, 2011). Toda esta funcionalidad habilita PSAD para el envío de alertas en ataques a nivel de aplicación. PSAD genera correos electrónicos en eventuales actividades maliciosas reportadas por la herramienta e incluye la ip origen del scanning, la cantidad de paquetes enviados a cada puerto, cualquier firma TCP, UDP o ICMP que hayan sido identificadas (e.g. "NMAP XMAS scan"), el rango de puertos bajo scan, el nivel de peligro (de 1 a 5), información dns reversa e información whois.

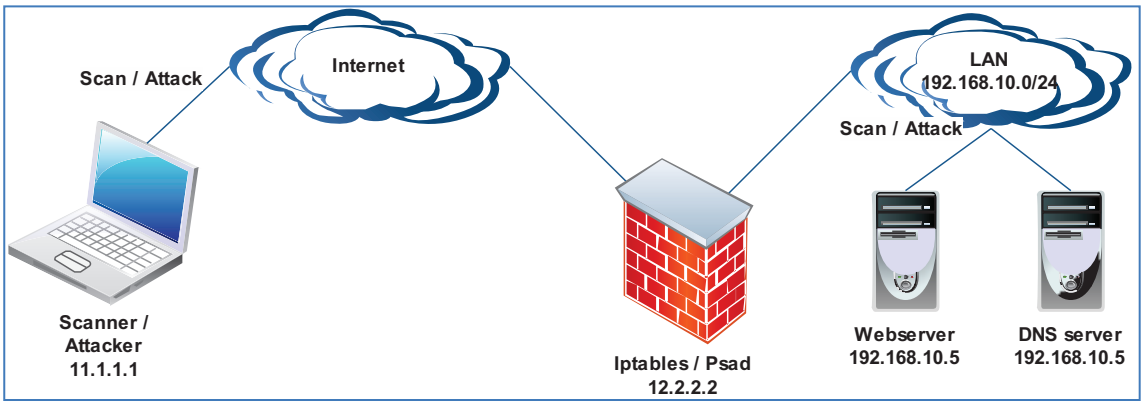

Figura 13. PSAD

2.1.7.6 NESSUS

Proyecto iniciado en 1998, cuando Renaud Deraison quiso que la comunidad de Internet tenga un escáner remoto de seguridad libre. Esta aplicación de escaneo de vulnerabilidades es desarrolladla para que interactúe en diversos sistemas operativos y consiste en un demonio (nessusd), que realiza el escaneo en el sistema objetivo, y consiste nessus, el cliente (basado en consola o gráfico) que muestra el avance e informa sobre el estado de los escaneos. Desde consola nessus puede ser programado para hacer escaneos programados con cron (Figura 14).

En operación normal, nessus comienza escaneando los puertos con nmap o con su propio escaneador de puertos para buscar puertos abiertos y después intentar varios exploits para atacarlo. Las pruebas de vulnerabilidad, disponibles como una larga lista de plugins, son escritos en NASL (NessusAttack Scripting Language, Lenguaje de Scripting de Ataque Nessus por sus siglas en inglés), un lenguaje scripting optimizado para interacciones personalizadas en redes. Opcionalmente, los resultados del escaneo pueden ser exportados como informes en varios formatos, como texto plano, XML,

HTML, y LaTeX. Los resultados también pueden ser guardados en una base de conocimiento para referencia en futuros escaneos de vulnerabilidades.

Por este motivo, Nessus forma parte de la repotenciación del sistema firewall ampliando las capacidades del mismo en aspectos de escaneo y análisis de seguridad de los sistemas implementados en la RED.

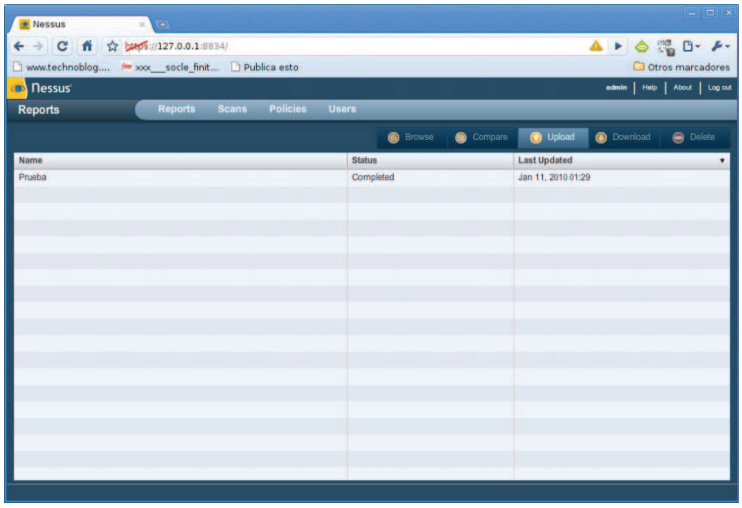

Figura 14. Interfaz GUI Nessus

Fuente: Imagen capturada en el desarrollo proyecto - NESSUS

2.1.8 DESARROLLO WEB

ClearOs Es una plataforma de servidor basada en Linux que se gestiona a través de una herramienta de administración en entorno web. Aunque se pueden implementar muchas de las características encontradas en ClearOS utilizando un servidor de uso general, como Ubuntu Server, este último proceso requiere una gran cantidad de conocimientos y experiencia.

ClearOS es una distribución Linux independiente, la mayoría del código fuente proviene de Red Hat Linux Enterprise. Se han añadido herramientas como detección de intrusiones (Snort) y Prevención (Snort sam), más de una docena de los paquetes de análisis de upstream han sido parchados.

Es una aplicación web la cual se encuentra desarrollada en PHP y contiene el siguiente framework:

- Framework MVC: **Codelgniter**
- Librería Javascript: **JQuery**
- Widgets para usuario: **JQueryUI**
- Widgets para usuario móvil: **JQuery Mobile**

2.1.8.1 Aplicaciones Web

Una aplicación web es un conjunto de páginas web que interactúan entre sí, con el usuario y con diversos recursos en un servidor web, incluidas bases de datos. En la Figura 15 se muestra la arquitectura de una aplicación web (Macromedia, Inc, 2002.).

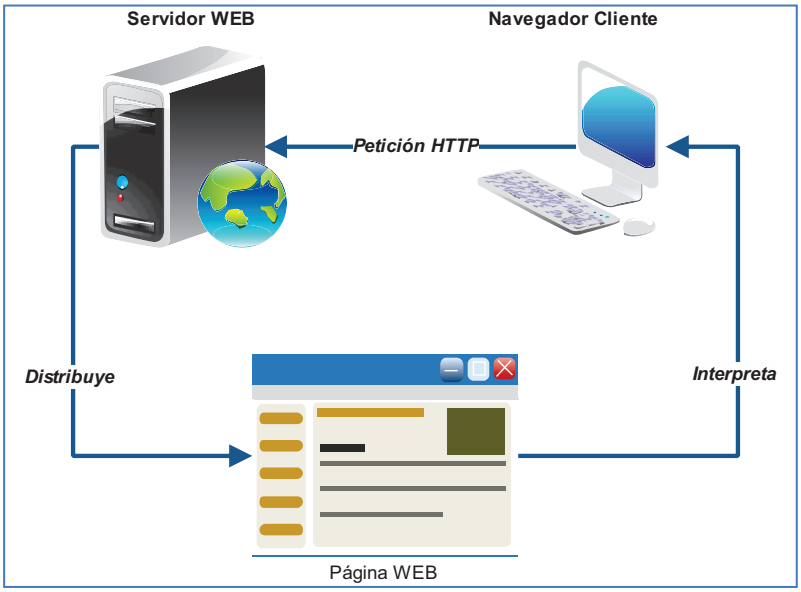

Figura 15. Arquitectura de una aplicación Web

a. Página Web

Es el resultado en hipertexto[1] o hipermedia[2] que proporciona un navegador de la WWW después de obtener la información solicitada. Su contenido puede ir desde un texto corto a un voluminoso conjunto de textos, gráficos estáticos o en movimiento, sonido, entre otros (Xpress Hosting, 2008).

b. Funcionamiento de aplicaciones Web

El funcionamiento de la aplicación depende básicamente de las páginas web que la conforman, estas pueden ser: estáticas y dinámicas. Las páginas creadas únicamente con HTML son básicamente estáticas, es decir, siempre muestran la misma información y no ofrecen ningún grado de interactividad con el usuario. Los únicos elementos de HTML que podrían de alguna manera considerarse interactivos son los formularios a través de los cuales se solicita información al usuario. Por otra parte cuando se requiere aumentar el dinamismo e interactividad de las páginas, es necesario recurrir a otros lenguajes y tecnologías para la creación de páginas dinámicas (Cobo, A. y Gómez, P., 2005).

c. Tecnologías para el desarrollo de aplicaciones web

Estas tecnologías se dividen en dos grupos: tecnologías de programación del lado del cliente y tecnologías de programación del lado del servidor. Del lado del cliente se emplean tecnologías como:

[1]Es una escritura no secuencial, un texto que se bifurca y que permite al lector elegir su trayectoria y su discurso, principalmente sobre una pantalla electrónica (6).
[2]Es una ampliación de hipertexto. Hipermedia le permite buscar y manipular formas de datos en multimedia; esto es, gráficas, sonido, video e información alfanumérica (5).

JavaScript, lenguaje interpretado basado en guiones que son integrados directamente en el código HTML. El código es transferido al cliente para que éste lo interprete al cargar la página.

Java, lenguaje de programación clásico en cuanto a que requiere un proceso de compilación. El código compilado puede ser integrado en la página web para que sea ejecutado por el cliente.

VBScript, al igual que JavaScript, un lenguaje basado en guiones que permite integrar programas directamente en el código HTML. Admite un doble uso, por un lado como lenguaje del lado del cliente, pero también como lenguaje del lado del servidor para la generación de páginas ASP.

A continuación se presentan algunas de las tecnologías de programación del lado del servidor más conocidas:

Programación CGI, son las siglas de Common Gateway Interface (Interfaz de Pasarela Común) lo que define es un estándar para establecer la comunicación entre un servidor web y un programa.

ASP (Active Server Pages), Páginas de Servidor Activas, tecnología diseñada por Microsoft para facilitar la creación de sitios web con una mayor sencillez que la empleada en la programación CGI. El principal inconveniente es la fuerte dependencia del entorno Microsoft, ya que requiere un servidor web bajo ambiente Microsoft.

Cold Fusion, tecnología que pertenece a Macromedia. Es una herramienta sencilla de aprender y bastante potente que funciona sobre la mayoría de los servidores web.

d. Sitio web

Conjunto de páginas web que comparten un mismo tema e intención y que generalmente se encuentra en un sólo servidor. Punto de la red con una dirección única y al que pueden acceder los usuarios para obtener información (Xpress Hosting, 2008).

e. Sistema cliente - servidor

En los sistemas cliente-servidor se distribuye el proceso de una aplicación entre múltiples ordenadores (de forma transparente para el usuario) en una red LAN o WAN. Los ordenadores suministrarán los datos comunes o compartidos a dicha aplicación o sistema. En una aplicación cliente/servidor, los programas GUI (interfaz gráfica de usuario para entradas y salidas) funcionan normalmente en el propio PC o en la estación de trabajo inteligente del usuario, que recibe el nombre de cliente. Otras partes del mismo programa de aplicación (como las lecturas y escrituras en las bases de datos) se realizan en otros elementos informáticos denominados servidores (Whitten J., Bentley, L. y Barlow, V., 2004).

Se da cuando existen múltiples servidores de procesamientos de bases de datos y cada uno de ellos procesa una base de datos diferente (Kroenke, 2003.).

La finalidad principal de los sistemas cliente/servidor es apoyar el desarrollo y la ejecución de las aplicaciones de bases de datos. Estos sistemas tienen una estructura muy sencilla de dos partes, las cuales consisten en un servidor (también denominado parte dorsal o servicios de fondo) y un conjunto de clientes (también llamados partes frontales, aplicaciones para el usuario o interfaces). En la Figura 19, se muestra la arquitectura de un sistema cliente/servidor (Date S., 2001).

f. Características de los sistemas cliente/servidor

Los sistemas cliente/servidor incluyen las siguientes características:

1) Desarrollo basado en GUI.

2) Un constructor GUI que soporte múltiples interfaces.

3) Desarrollo orientado a objetos con soporte para la reutilización de código.

4) Un diccionario de datos con un depósito central de datos y aplicaciones.

5) Soporte de múltiples bases de datos (relacional, red, jerárquica y archivo simple).

6) Acceso a los datos sin importar el modelo.

7) Acceso perfecto a múltiples bases de datos.

8) Soporte al desarrollo de equipos.

9) Creación de prototipos y capacidades de desarrollo rápido de aplicaciones.

10) Soporte de múltiples plataformas.

11) Soporte de múltiples protocolos.

La Figura 16 muestra claramente los componentes que conforman una arquitectura cliente-servidor.

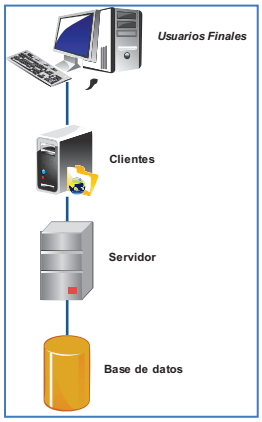

Figura 16. Arquitectura Cliente-Servidor

2.1.8.2 PHP (HyperText Preprocessor)

Es un lenguaje de *scripting* que permite la generación dinámica de contenidos en un servidor web. Entre sus principales características se pueden destacar su potencia, alto rendimiento y su facilidad de aprendizaje. PHP es una eficaz herramienta de desarrollo para los programadores web, ya que proporciona elementos que permiten generar de manera rápida y sencilla sitios web dinámicos (Peña, 2 de agosto de 2012).

PHP comenzó como un simple lenguaje de scripts. A medida que las versiones avanzaban, se fueron incluyendo algunas características que permitían programar con orientación a objetos. Con la aparición de PHP 5, los desabolladores tenemos una verdadera plataforma de programación orientada a objetos, gracias a la potencia del nuevo engine Zend 2.0, que permite ejecutar más rápido y eficientemente este tipo de programas.

La programación orientada a objetos utiliza los elementos clase y objeto con punto de inicio. Actualmente, en las carreras técnicas, es posible encontrar enseñanzas sobre lenguajes orientados a objetos como Java o C++.

La programación orientada a objetos desarrolla nuevos conceptos y terminologías que deben aprenderse correctamente para generar buen código.

a. Definición de clases

Una clase es un tipo de dato que contiene, en una misma estructura, variables y funciones. Una clase es una especie de plantilla desde la que los objetos son instanciados y toman su valor. Desde una clase se pueden construir varios objetos del mismo tipo. La definición es la siguiente:

```
class pagina_Web
{
	var $titulo;
	function get Titulo()
	{
		return $this->titulo;
	}
}
```

Mediante la palabra reservada **class** definimos una clase completa. Utilizando var podemos definir las variables que utilizaremos en el desarrollo del programa. La palabra reservada var desaparecerá en versiones siguientes de PHP, a favor de las palabras public, prívate y protected. Las funciones se definen como métodos, siempre que estén dentro de la construcción de la clase. A continuación vamos a definir una clase completa cuya misión será mostrar una página Web. Podemos seguir con el ejemplo ampliando las funciones:

```
<?php
	class pagina_Web
	{
	var $titulo;
	function set Titulo($titulo = "Titulo por defecto")
	{
		$this->titulo = $titulo;
	}
	function get Titulo()
	{
		return $this->titulo;
	}
```

```
        function cabecera()
        {
                echo ("<htmlxheadxtitle>") ;
                echo $this->titulo;
                echo (" </titlex/headxbody>") ,•
        }
        Function cuerpo()
        {
                echo("Este es el cuerpo de la página Web");
        }
        function pie ( )
        {
                echo ("</bodyx/html>") ;
        }
        function mostrar_pagina()
        {
                echo $thís->cabecera () ;
                echo $this->cuerpo() ;
                echo $this->pie();
        }
}
```

Como puede observar, la clase se divide en varias funciones y en una sola variable.

La variable $titulo guardará el título dé la página Web. Las funciones cabecera(), cuerpo() y pie() son las encargadas de mostrar las distintas partes de una Web. Estas funciones se llaman desde la función mostrar_pagina(). Es interesante observar que, en unas pocas líneas de código, se ha definido un objeto que podría utilizarse como base de muchas aplicaciones. El operador $this es una variable que contiene el objeto actual

desde el que la invocamos. Para acceder dentro de un objeto a funciones o variables propias, debe anteponerse al nombre de éstas la expresión $this - >.

Para acceder desde la función mostrar_pagina() a cualquiera de las funciones de la clase tendrá que escribir:

```
$this->cabecera ();
$this->cuerpo ();
$this->pie ();
```

b. Instancia de clase

Para que el código funcione, necesita crear el objeto. Esto se hace utilizando el operador new seguido del nombre de la clase:

```
$pagina = new pagina_Web() ;
```

Con el código anterior hemos creado un objeto que contiene todas las variables y funciones de la clase pagina_Web ( ). La variable que contiene ese objeto es $pagina. Para acceder a las funciones desde el nuevo objeto creado tenemos que utilizar la variable que contiene al objeto, en este caso $pagina seguido del operador - > y el nombre de la función. Para mostrar la Web tenemos que seguir estos sencillos pasos:

```
$pagina = new pagina_Web();
$pagina->setTitulo("Página Web nueva");
$pagina- >mostrar_pagina ();
```

c. Función constructor

Existen algunas funciones especiales en la definición de una clase. La más importante es el constructor. Esta se ejecuta cada vez que se crea un nuevo objeto y

permite crear las variables iniciales que se necesitan, como el título de la Web o el autor. El nombre de la función debe ser construct() . Nuestro constructor se encargará de crear el título de la página Web que estamos creando:

```
function construct($titulo)
{
        $this->set Titulo($titulo);
}
```

d. Herencia

La programación orientada a objetos permite heredar de otras clases. Con esta técnica puede ahorrar mucho tiempo de trabajo. La clase hija (clase que hereda de otra) adquiere estas propiedades:

- Automáticamente obtiene todas las variables miembro de la clase padre.
- También obtiene todas las funciones miembro de la clase padre, que funcionarán exactamente de la misma forma.
- La clase hija puede a su vez definir nuevas variables y funciones.

La sintaxis es la siguiente:

```
class pagina_Web_formulario extends pagina_Web
{
        function formulario_inicio()
        {
        //Escribir el código necesario
        }
}
```

La palabra reservada **extends** indica que la nueva clase creada será una extensión (heredará) de la clase que se escribe justo a la derecha de la definición.

Puede crear un objeto formulario a partir de la clase pagina_Web de la siguiente forma:

```
class pagina_Web_formulario extends pagina_Web
function formularío_inicio($accion)
{
        echo ("<formaction=\"$accion\">");
        }
        function formulario_fin()
        {
        echo ("</form>");
        }
        function formulario_caja_texto($nombre)
        {
        echo ("$nombre<input type=\"text\" name=\"$nombre\">");
        }
        function formulario_boton()
        {
        echo ("<input type=\"submit\" name=\"Submit\"
        valúe = \"Enviar\">" ) ;
        }
        function mostrar_pagina ()
        {
        $this->cabecera () ;
        $this- >formulario_inicio (" Índex .php") ;
        $this- >formulario_caja_texto ("Nombre") ;
        $this->formulario_boton();
        $this->formulario_fin();
        $this- >pie () ;
        }
}
```

Si ahora ejecuta el código instanciando la clase que hemos creado, puede ver que las funciones cabecera() ,pie() ,incluso el constructor pueden ser utilizadas aun no perteneciendo a la clase pagina_Web_formulario.

```
$formulario  = new pagina_Web_formulario{"Pagina con formulario");
$formulario->mostrar_pagina ( );
```

e. Métodos o funciones de objeto

Cuando definimos una clase hija, las funciones de la clase padre son automáticamente heredadas. Se llama redefinición de métodos a la creación de funciones en la clase hija, con el mismo nombre que en la clase padre.

En el ejemplo anterior, la función mostrar_pagina() de la clase pagina_Web_formulario está redefinida, ya que existe una función con el mismo nombre en la clase pagina_Web. Al instanciar el objeto y ejecutar el método mostrar_pagina( ) , el método que se ha definido es la clase hija.

PHP 5, con su nuevo motor ZendEngine 2, introduce la palabra reservada final.

Si en la clase padre ponemos delante de cualquier función la palabra final, ésta función no podrá ser sobrecargada en las clases que la hereden.

También podemos declarar clases completas como final, lo que significará que no podrán ser heredadas.

```
final function mostrar_pagina()
{
        echo $this->cabecera();
        echo $this->cuerpo();
        echo $this->pie();
}
```

Si definimos la función anterior como final en la clase pagina_Web, la clase pagina_Web_formulario no podrá tener una función con este mismo nombre y mostrará un error en pantalla (Cabezas, 2004).

CAPITULO III

EVALUACIÓN DE SISTEMA FIREWALLING

3.1 SELECCION, INSTALACION Y CONFIGURACION DE SISTEMAS OPEN SOURCE

Una apropiada selección del Sistema Firewalling puede simplificar de forma significativa, los recursos de Hardware, implementaciones adicionales, soporte técnico, actualizaciones de seguridad en general, mitiga potenciales riesgos de seguridad de la información.

Al analizar, evaluar, seleccionar y repotenciar una plataforma de firewalling de código abierto basado en las características distintivas de un sistema propietario y verificando su comportamiento en términos de performance y efectividad ante ataques informáticos, reforzaría las seguridades informáticas, fortaleciendo cada vez más el concepto de LINUX como una potente herramienta en aspectos de seguridad y confiabilidad (Figura 17).

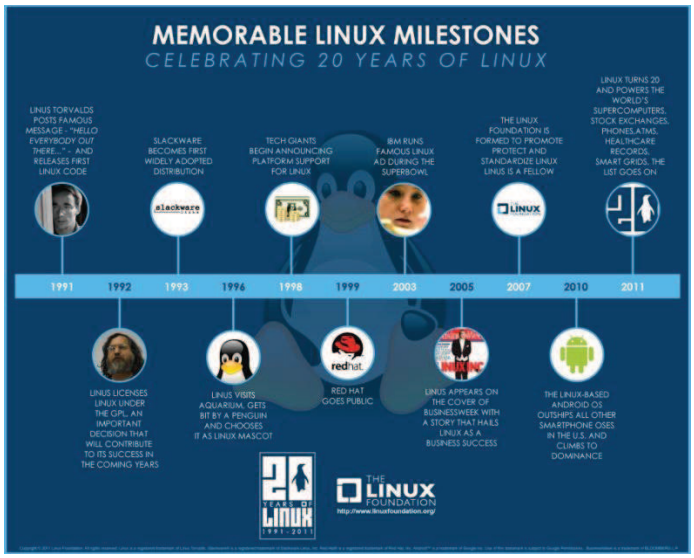

Figura 17. Alcances LINUX
Fuente: www.linuxfoundation.org

3.1.1 SELECCIÓN DE SISTEMAS OPEN SOURCE

A continuación se desglosa un listado parcial de FIREWALLS Open Source cuyo método de búsqueda fue su sistema operativo y soporte.

FIREWALL	ESTADO	SISTEMA BASE	SOPORTE
Alpine Linux	Active	Distribución Linux	Disponible en foros públicos
BSD Router Project	Active	Derivado de FreeBSD	Disponible en foros públicos
Bifrost Network Project	Active	Distribución Linux	Disponible en foros públicos
ClearOS	Active	Derivado de RedHat	Gratuito con servicios limitados - requiere suscripción
Devil-Linux	Active	Distribución Linux	Disponible en foros públicos
DD-WRT	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción
Endian Firewall	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción
Floppyfw	Active	Distribución Linux	Disponible en foros públicos
FREESCO	Active	Distribución Linux	Disponible en foros públicos
FreeWRT	Active	Distribución Linux	Disponible en foros públicos
Gibraltar	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción

Global TechnologyAssociates, Inc.	Active	Derivado de FreeBSD	Disponible en foros públicos con servicios limitados - requiere suscripción
Halon Security	Active	Derivado de OpenBSD	Disponible en foros públicos con servicios limitados - requiere suscripción
IPCop	Active	Distribución Linux	Disponible en foros públicos
IPFire	Active	Distribución Linux	Disponible en foros públicos
LEAF Project	Active	Distribución Linux	Disponible en foros públicos
MikrotikRouterOS	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción
m0n0wall	Active	Derivado de FreeBSD	Disponible en foros públicos
OpenWrt	Active	Distribución Linux	Disponible en foros públicos
Openwall	Active	Distribución Linux	Disponible en foros públicos
pfSense	Active	Derivado de FreeBSD	Disponible en foros públicos
SmartRouter	Active	Distribución Linux	Disponible en foros públicos
Simplewall	Active	Distribución Linux	Disponible en foros públicos
SME Server	Active	Derivado de CentOS	Disponible en foros públicos
Smoothwall	Active	Distribución Linux	Disponible en foros públicos con servicios limitados -

			requiere suscripción
Sphirewall	Active	Distribución Linux	Disponible en foros públicos
Threenix	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción
Untangle	Active	Derivado de Debian	Disponible en foros públicos con servicios limitados - requiere suscripción
Vyatta	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción
Zeroshell	Active	Distribución Linux	Disponible en foros públicos con servicios limitados - requiere suscripción
Zentyal (formelyeBoxPlatform)	Active	Derivado de Ubuntu	Disponible en foros públicos con servicios limitados - requiere suscripción

Tabla 2. Listado Firewalls OpenSource

En la presente investigación se tomaron aspectos técnicos basados en la herramienta propietaria y diferenciados por su sistema operativo, cabe mencionar que los sistemas escogidos son los que mejor se adaptaron a las necesidades técnicas, con soporte avanzado y mejor experiencia en el campo de las seguridades e inclusive programas de certificación.

3.1.1.1 CLEAROS - REDHAT

Este sistema operativo es considerado uno de los líderes por su eficiencia, escalabilidad y fiabilidad a la hora de gestionar sus recursos, con alta seguridad e integridad de sus datos y programas de certificación.

3.1.1.2 ZENTYAL - UBUNTU

Posee recursos gubernamentales para su desarrollo (Gobierno de Aragon), y mantiene continuos programas de capacitación y administración de la Herramienta y oficialmente UBUNTU lo acredita.

3.1.1.3 PFSENSE - FREEBSD

Tiene como soporte a los desarrolladores de FREBSD. De las herramientas investigadas, bajo este sistema operativo, es la que cuenta con mayor soporte WEB y herramientas que se adaptan a la investigación.

WATCHGUARD - HERRAMIENTA PROPIETARIA	**CLEAROS**	**PFSENSE**	**ZENTYAL**
Sistema Operativo	REDHAT	FREEBSD	UBUNTU
Implementación de políticas de seguridad capa de aplicación (capa 7) (proxy transparente).	SI	SI	SI
Políticas de seguridad en capa de aplicación pre-configuradas para protocolos comunes: http, https, POP3, SMTP, FTP, TFTP, DNS, SIPv2, H323.	SI	SI	SI
Creación de perfiles de usuario.	SI	SI	SI

Soporte de VPN.	SI	SI	SI
Filtraje de Contenido	SI*[3]	SI*	SI*
Filtraje de Contenido Categorizado	SI	SI	SI
Antivirus y antispam.	SI	SI	SI
Actualización del Antivirus y antispam automática	SI*	SI	SI*
Soporte de bloqueo de Spyware.	SI*	SI	SI*
IPS	SI	SI	SI
Actualización automática del IPS.	SI*	SI*	SI*
NAT	SI	SI	SI
Capacidades de Networking	SI	SI	SI
Control de ancho de banda	SI	SI	SI
Interfaz de administración gráfica en tiempo real.	SI	SI	SI
Monitoreo.	SI	SI	SI
Logs y reportes	SI	SI	SI
Base de datos SQL.	SI	SI	SI

Tabla 3. Comparativa WatchGuard vs herramientas OpenSource

3.1.2 DISEÑO E IMPLEMENTACIÓN DE LA TOPOLOGÍA DE PRUEBA

Las Figuras 18 y 19, muestran el diseño de la topología de pruebas. Como se puede apreciar, se diseñó e implementó un escenario de red LAN y WAN separado por un firewall, que fue configurado en un ambiente real y validado en un entorno virtual, mediante tres sistemas de software: Pfsense, Zential y ClearOS. Este laboratorio

[3] Requiere configuración adicional - misma que será ampliada en la sección de configuración.

permitió efectuar las diferentes evaluaciones a los firewalls seleccionados en un entorno real aplicando un esquema de pruebas igual para sus distintas distribuciones.

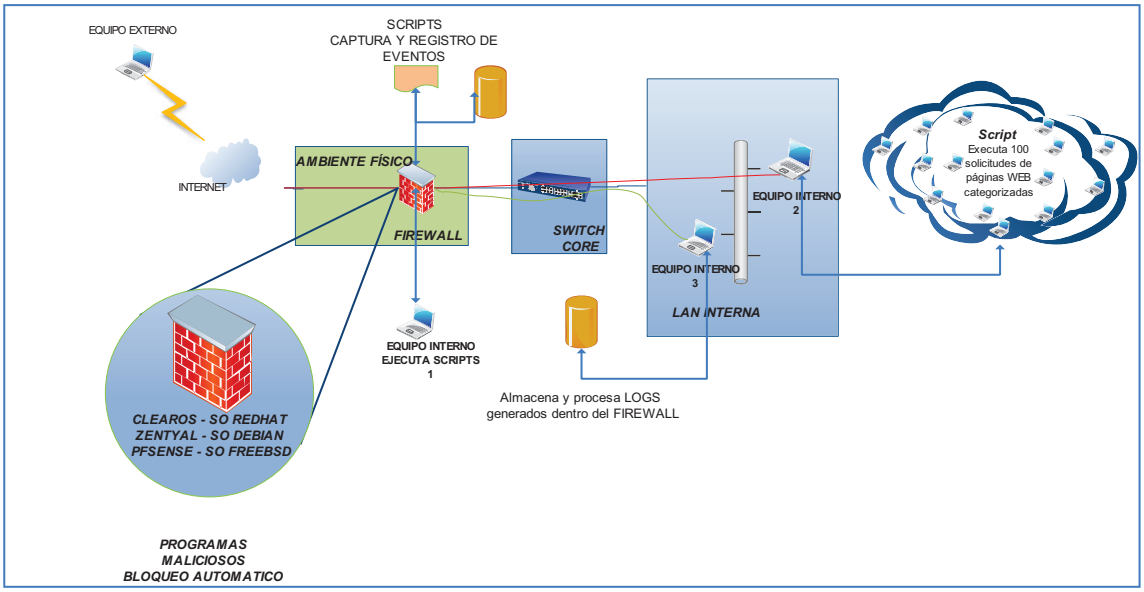

Figura 18. Diseño y topología de pruebas – ambiente real

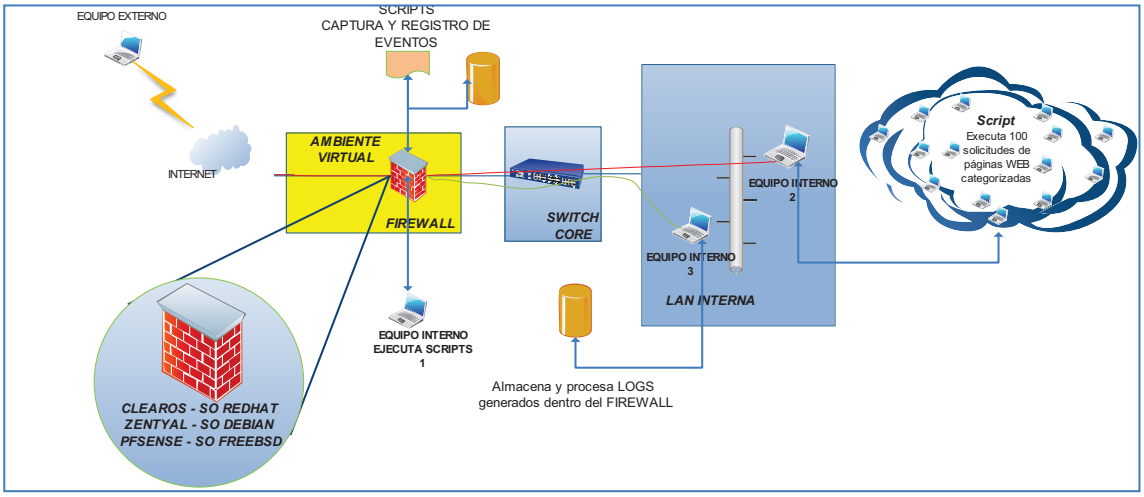

Figura 19. Diseño y topología de pruebas – ambiente virtual

Como parte del experimento, para realizar el análisis de los diferentes sistemas de firewalling, se instaló cada uno de ellos en una unidad de almacenamiento externo

(disco duro), con el propósito de brindar igualdad a los distintos escenarios de pruebas planificados, lo cual aprovisionó portabilidad, independencia, facilidad de montaje, desmontaje, flexibilidad y estabilidad en la evaluación, logrando de esta manera disminuir la pérdida de configuración del sistema de arranque del sistema operativo local y un respaldo completo del disco duro, para evitar reinstalaciones.

Durante esta investigación se contó con un firewall de caja negra firebox-core UTM WATCHGUARD 550E, cuya licencia anual contempla control de acceso a la Web, IPS/IDS, QoS, AntiSpam/Malware, virus, gusanos, troyanos, pishing, spyware y todo tipo de malware a nivel de frontera.

Dentro de la infraestructura de red LAN se organizó un esquema de direccionamiento IP dinámico DHCP, el cual fue asignado por el firewall, en un segmento de red en el rango 192.168.0. 10 a 192.168.0.20 para equipos internos.

3.1.3 INSTALACION DE SISTEMAS OPENSOURCE

Las instalaciones de Clear OS, Zentyal y Pfsense, por medio de CD, siguen parámetros establecidos por los sistemas base, REDHAT, UBUNTU y FREEBSD, respectivamente. Los datos solicitados al operador pueden variar desde el idioma, instalación asistida o en modo experto, o si la instalación es nueva o simplemente una actualización. Independientemente de la forma de instalación el usuario deberá tener en cuenta, aspectos técnicos que permitan la interacción y configuración entre un equipo de la red y el FIREWALL.

Las figuras (20, 21, 22) nos presentan las diferentes interfaces de instalación que despliegan Zentyal, Pfsense y ClearOs en su respectivo orden.

Figura 20. Instalación Zentyal

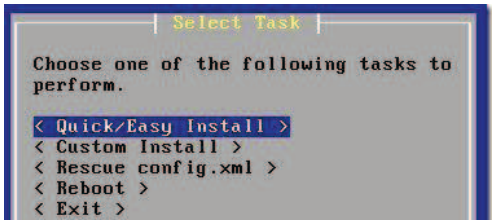

Figura 21. Instalación Zentyal

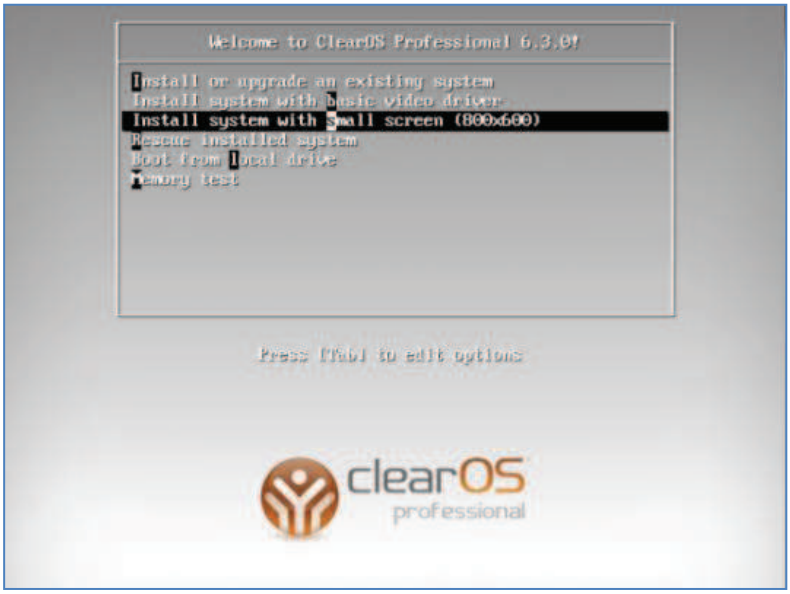

*Figura 22.*Instalación ClearOs

3.2 HARDWARE Y SISTEMA DE ARCHIVOS

Es recomendable que un administrador de seguridad dedique un equipamiento (HARDWARE) exclusivo para este servicio, evitando así particiones de disco innecesarias, ya que al ser un equipo de frontera no deberán ser interrumpidos sus servicios, salvo el caso de mantenimiento, actualización, cambio de hardware o re potenciación del mismo.

Por defecto las distribuciones ya incluyen un tipo de sistema de archivos de disco:

- CLEAROS(EXT4)
- ZENTYAL(EXT4)
- PFSENSE (UFS)

En algunos casos los sistemas permiten al usuario avanzado realizar cambios del sistema de archivos (CLEAROS Y ZENTYAL).

Las figuras 23 y 24 muestran como el asistente de instalación de ClearOs permite seleccionar entre una partición estándar o una partición tipo RAID.

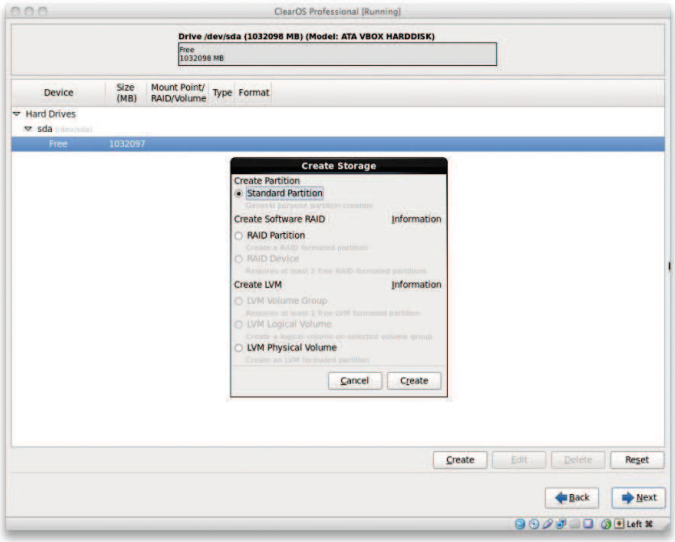

Figura 23. Configuración ClearOs

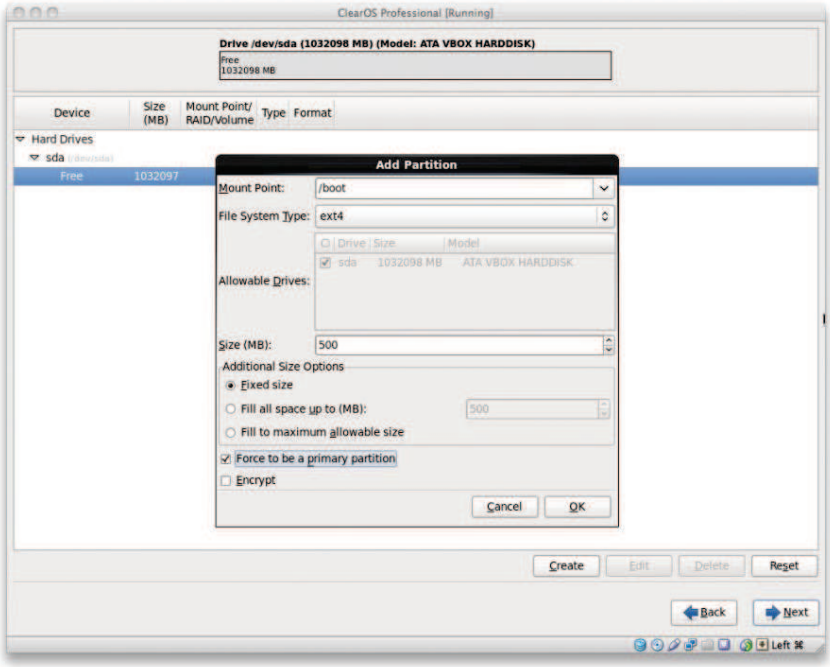

Figura 24. Configuración ClearOs

3.3 SEGURIDAD DEL SISTEMA - ADMINISTRADOR

Un aspecto muy importante que un administrador de seguridad debe integrar es el manejo de contraseñas y/o un nombre de administrador, dado que rara vez eligen contraseñas que sean difíciles de adivinar y fáciles de recordar (Security &Privacy, IEEE Date of Publication: Sept.-Oct. 2004 Author(s): Yan, J. Chinese Univ. of Hong Kong, China Blackwell, A.; Anderson, R.; Grant, A.) La convergencia de contraseña y/o usuario de estas herramientas pueden variar:

3.3.1 CLEAROS

Establece y habilita por defecto al usuario root (Figura 25), y en la instalación del sistema solicita una contraseña para el mismo, lastimosamente no realiza un control exhaustivo de la misma y permite pasar contraseñas sencillas (ej. 12345678).

3.3.2 ZENTYAL

Al ser su sistema base UBUNTU - DEBIAN, no habilita a root como usuario administrador, sino, solicita se establezca el nombre del usuario y su respectiva contraseña (figuras 26 y 27). Al igual que la primera herramienta no realiza un control exhaustivo de estos valores.

3.3.3 PFSENSE (UFS)

La concepción de seguridad de esta herramienta fue mucho más arriesgada al incluir un nombre de usuario y contraseña por defecto (Figura 28) sin ser un campo obligatorio su cambio inmediato.

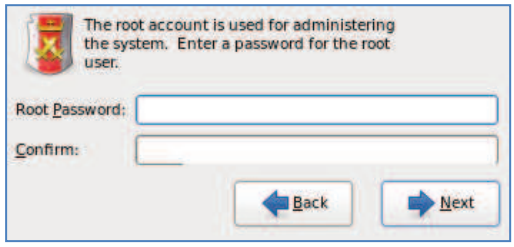

Figura 25. Ingreso sistema usuario root

Figura 26. Configuración usuario - Zentyal

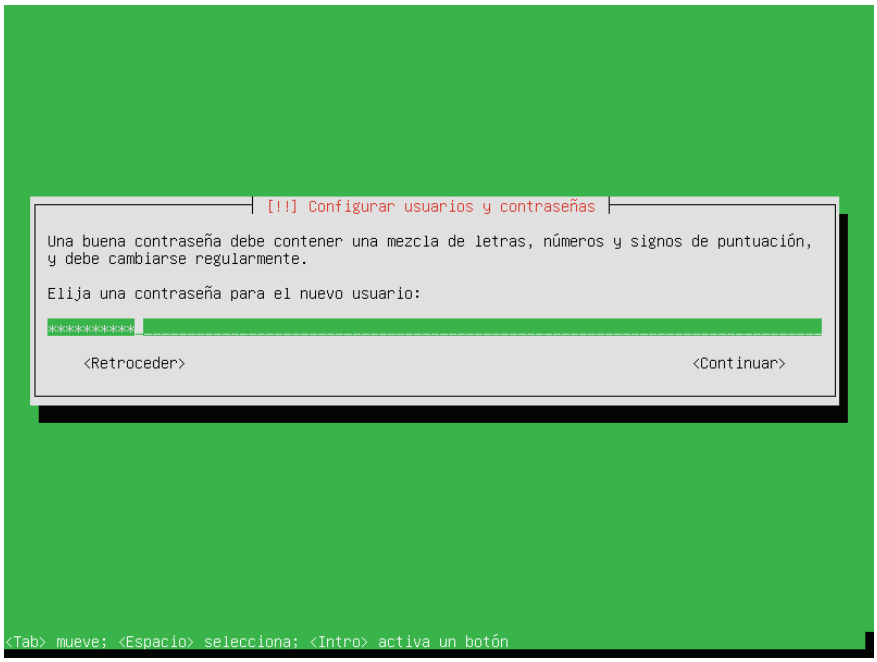

Figura 27. Configuración contraseña Zentyal

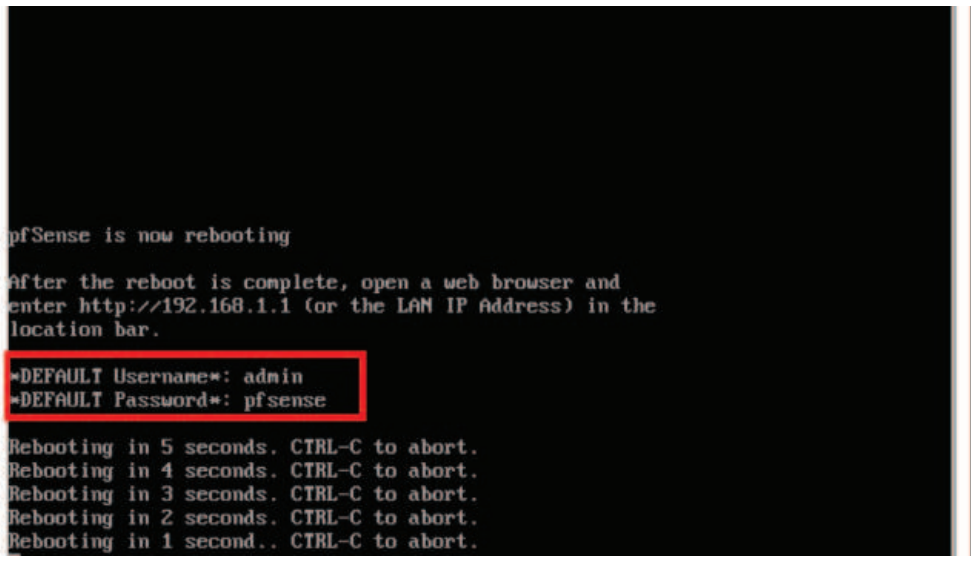

*Figura 28.*Configuración usuario – password Pfsense

3.4 DIRECCIONAMIENTO IP

El protocolo TCP/IP debe ser cuidadosamente configurado de modo que los dispositivos de red involucrados operaren de manera eficiente con el servidor (Internet Computing, IEEE Date of Publication: Jul/Aug 1999 Author(s): Droms, R. Comput.

Sci., Bucknell Univ., Lewisburg, PA Volume: 3, Issue: 4 Page(s): 45 - 53 Product Type: Journals & Magazines). Inicialmente la configuración de este valor en la instalación, en los sistemas FIREWALL, es manual y propensa a errores. El mecanismo de implementación del protocolo puede variar según la distribución, para el caso de CLEAROS y ZENTYAL se lo puede realizar en la instalación (figuras 29, 30). Al contrario de PFSENSE que una vez finalizada la misma habilita un menú, modo texto, donde puede configurarse la interfaz y su direccionamiento (Figura 31).

Figura 29. Configuración de red ClearOs

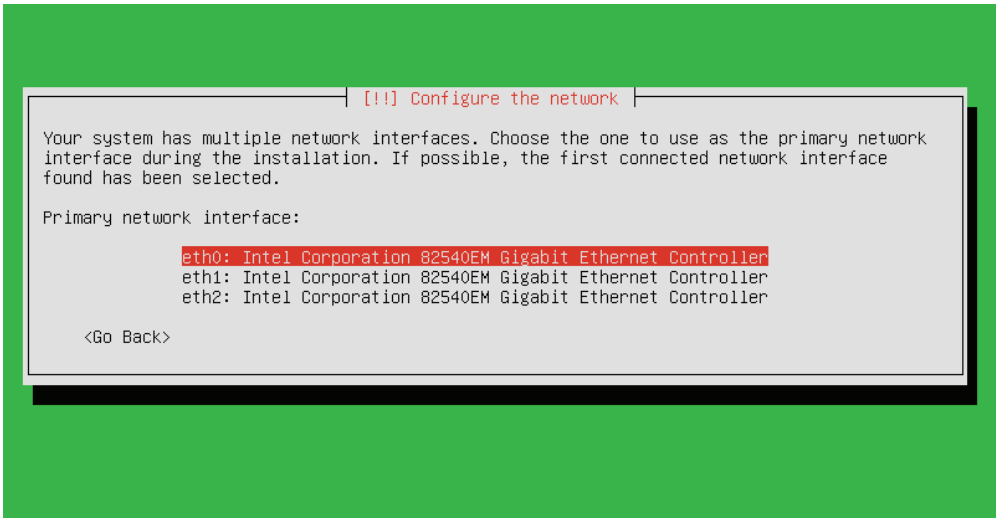

*Figura 30.*Configuración de red - Zentyal

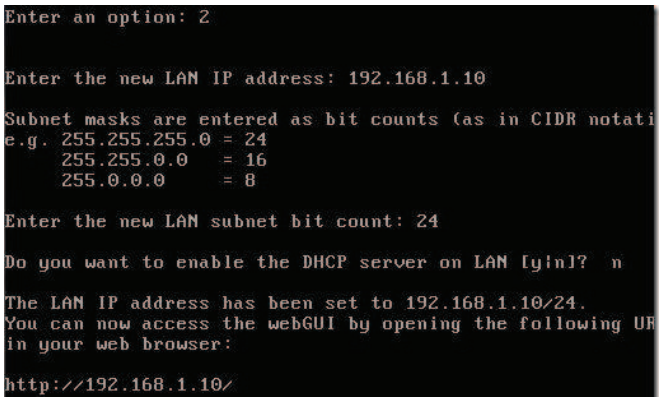

Figura 31. Configuración red Pfsense

3.5 CONFIGURACIÓN VÍA INTERFACE WEB DE USUARIO - GUI

La tendencia de las plataformas de seguridad OPEN SOURCE es proveer al administrador un método de configuración rápido y seguro por medio de interfaces gráficas de usuario (GUIs), el acceso a esta interfaces son parametrizables para que su acceso sea encriptado o no (https) e inclusive cambian su puerto WEB para una mayor seguridad.

Las figuras 32, 33 y 34 muestran las diferentes interfaces gráficas de usuarios para el ingreso a las distribuciones (ZENTYAL, CLEAROS y PFSENSE) respectivamente.

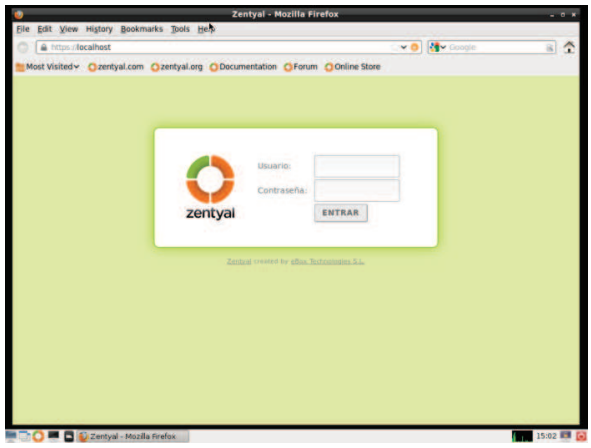

Figura 32. Interface Web - Zentyal

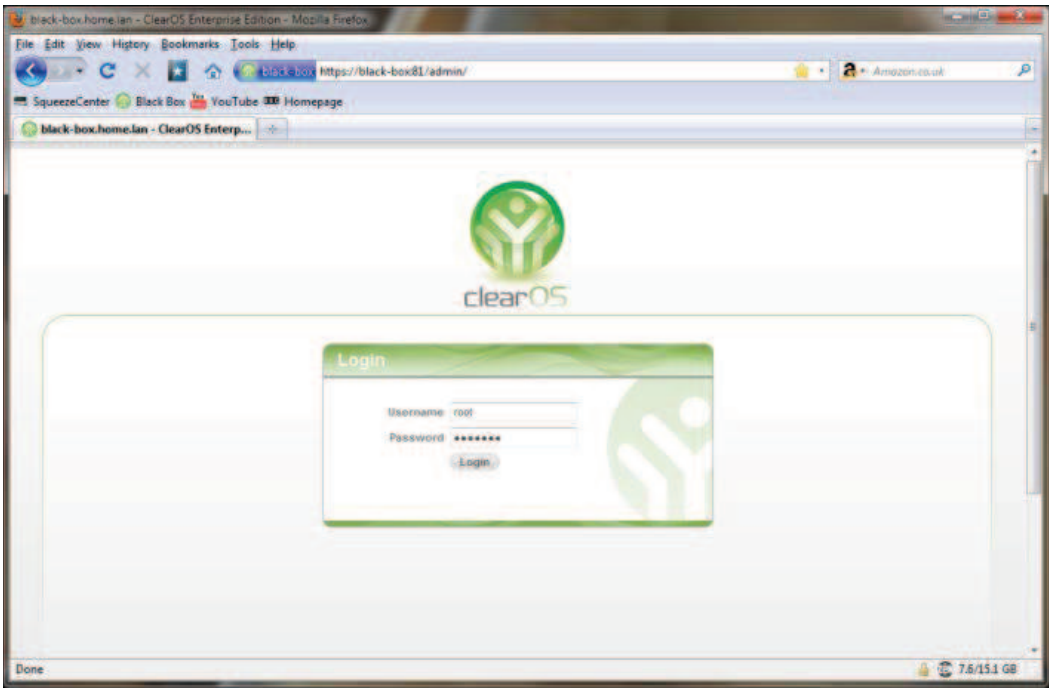

Figura 33. Interface Web - ClearOs

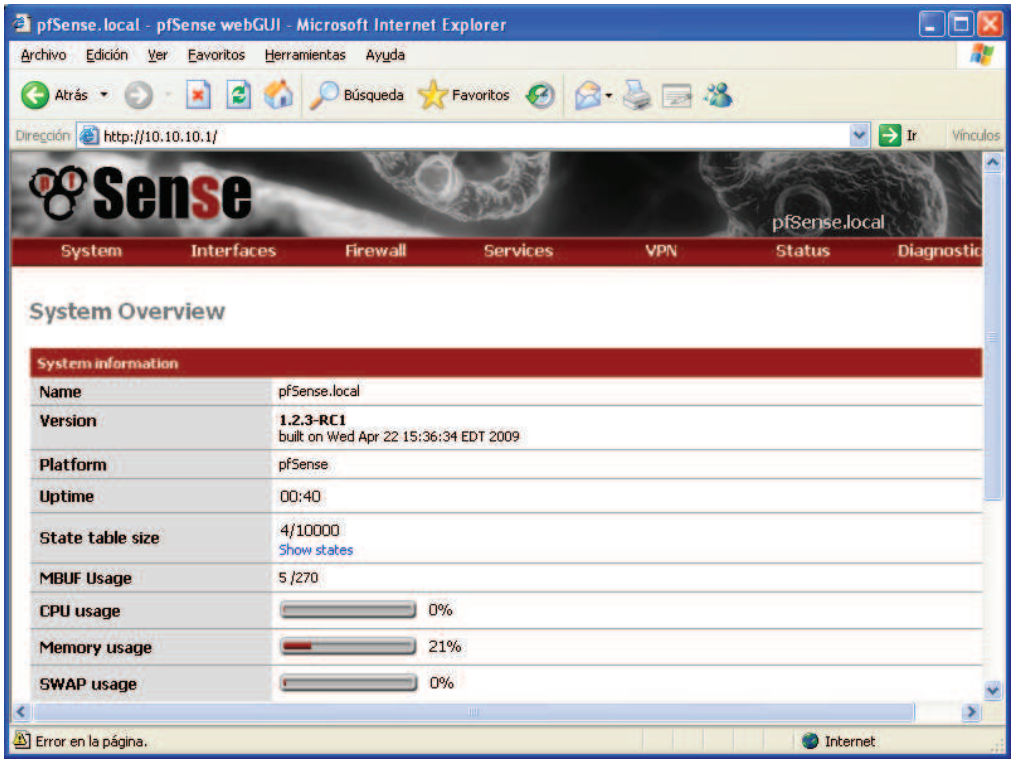

Figura 34. Interface Web –Pfsense

3.6 CONFIGURACIONES DE SISTEMAS

Watchguard al igual que las herramientas Open source (CLEAROS, PFSENSE y ZENTYAL), según sus potencialidades, deben ser configuradas de manera similar para obtener datos equivalentes bajo un entorno establecido:

3.6.1 RED Y DHCP

3.6.1.1 WATCHGUARD Y FIREWALLS OPENSOURCE

Las figuras 35, 36, 37, 38 y 39 muestran como WATCHGUARD al igual que los sistemas firewall Open Source dispone en su barra de tareas herramientas para configurar los parámetros relacionados con la Interfaz de Red.

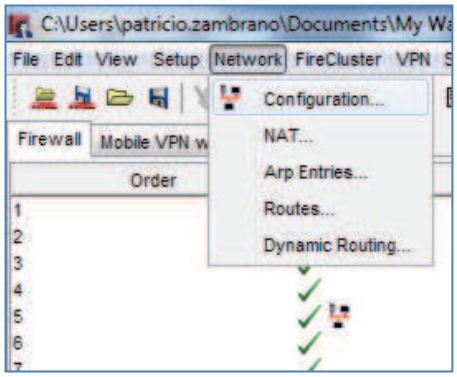

Figura 35. Configuración de red -WatchGuard

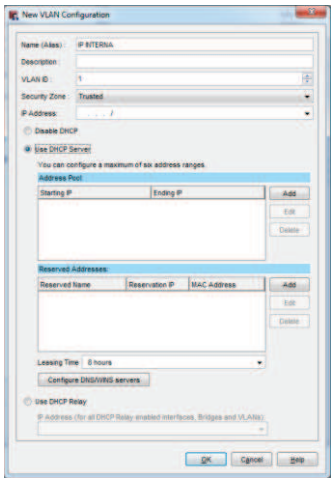

Figura 36. Configuración de red (DHCP) -WatchGuard

3.6.1.2 CLEAROS

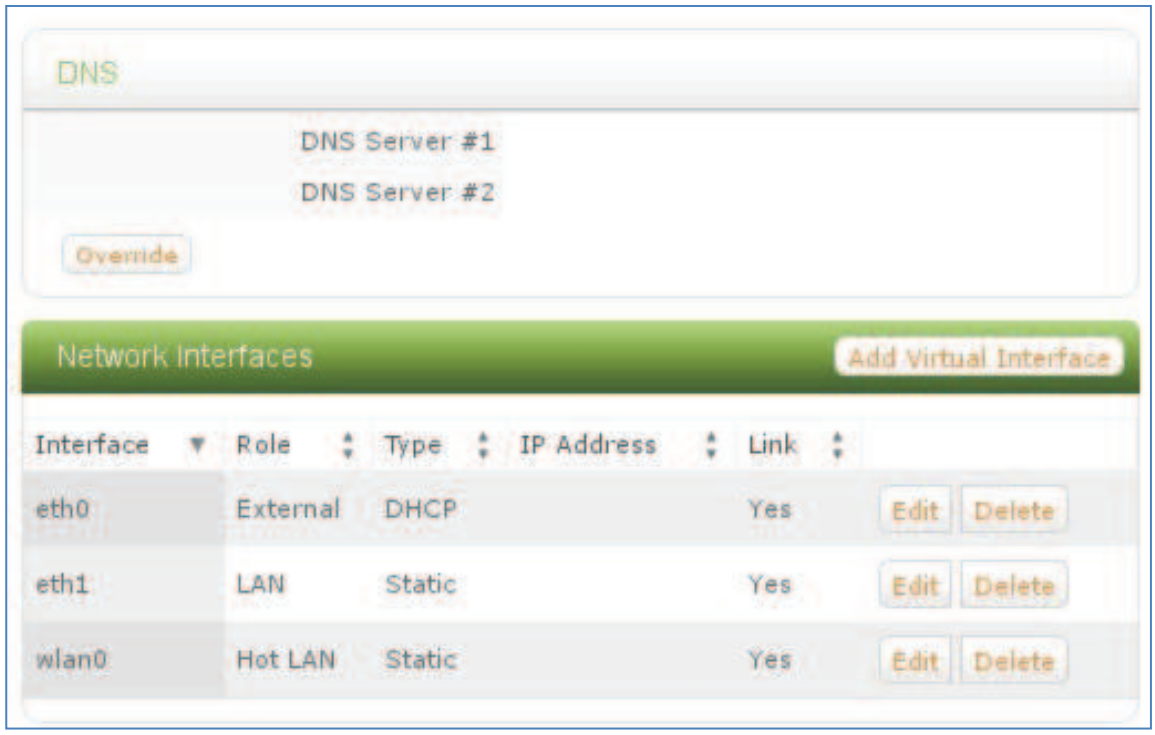

Figura 37. Configuración red – ClearOs

3.6.1.3 ZENTYAL

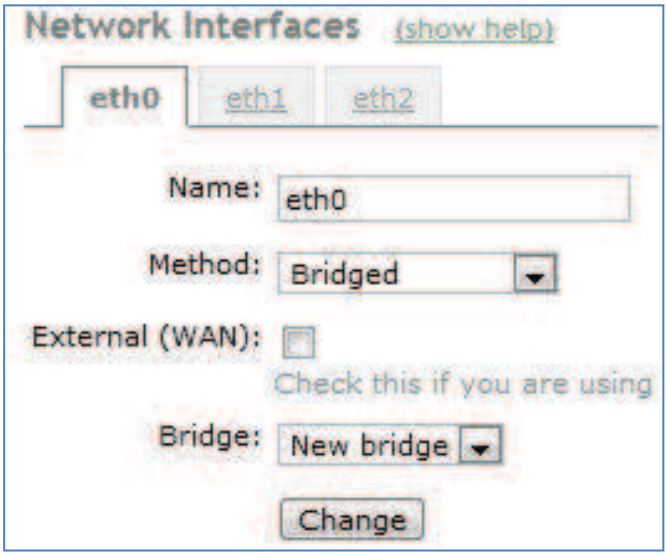

Figura 38. Configuración red - Zentyal

3.6.1.4 PFSENSE

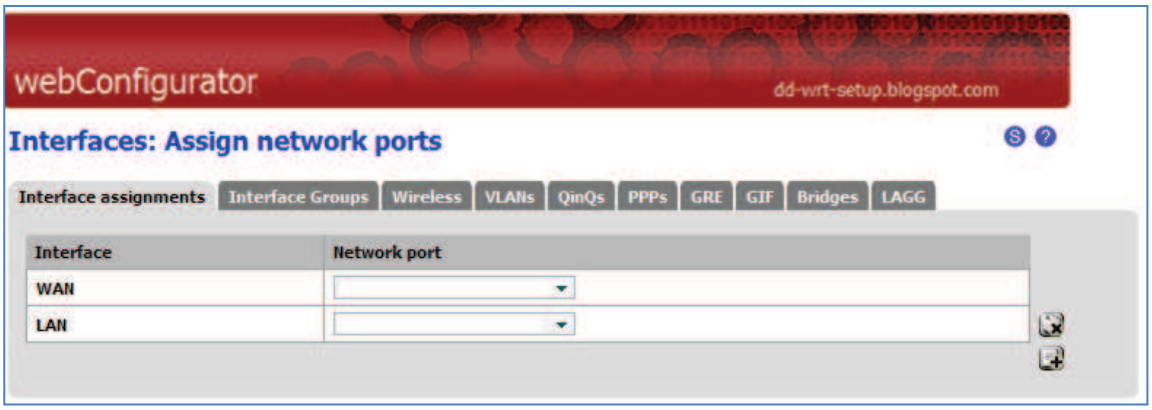

Figura 39. Configuración red - Pfsense

3.6.1.5 CONFIGURACIÓN NAT

Una de las técnicas que solucionan la limitante de direcciones públicas y su enrutamiento, es NAT (Network Address Translation), que permitirá conectar los equipos de la misma subred a Internet, utilizando únicamente una dirección IP pública para ello. Las figuras 40,41,42 y 43.

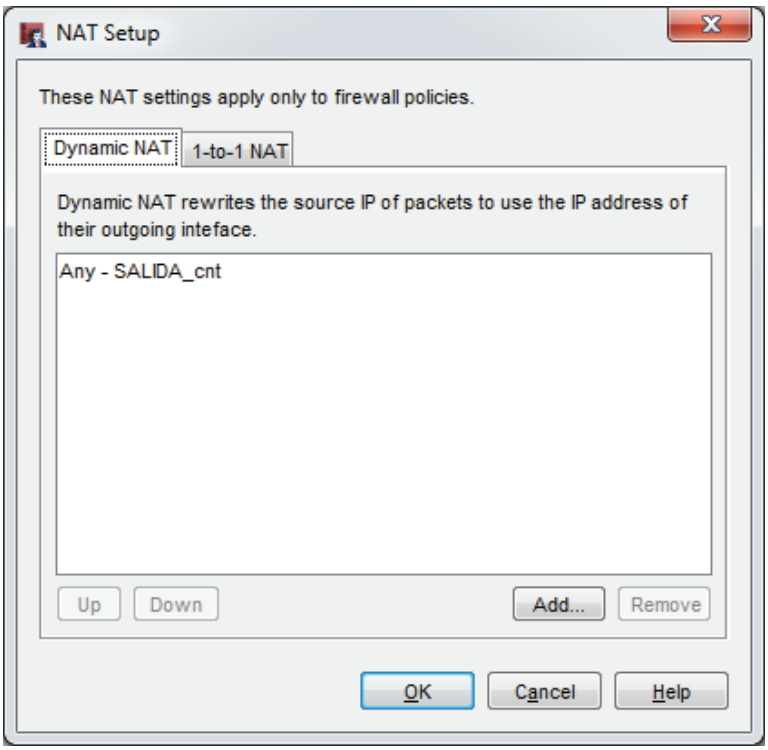

*Figura 40.*Configuración NAT

3.6.1.6 CLEAROS

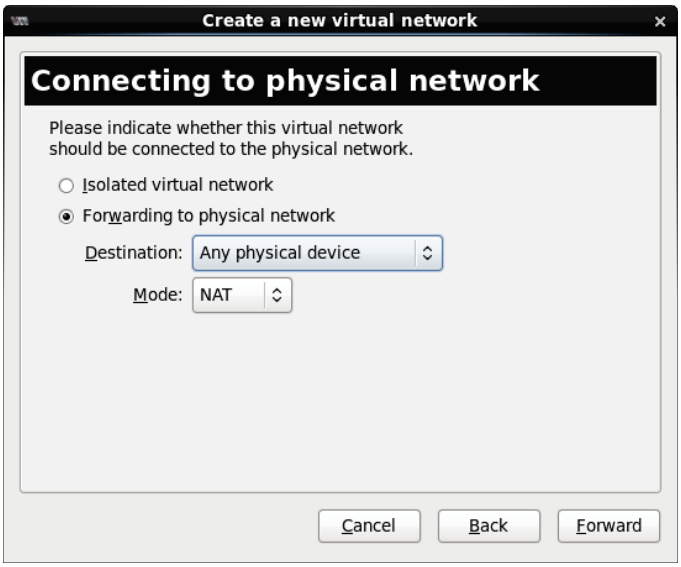

Figura 41. Configuración red virtual - ClearOS

3.6.1.7 ZENTYAL

Figura 42. Configuración red - Zentyal

3.6.1.8 PFSENSE

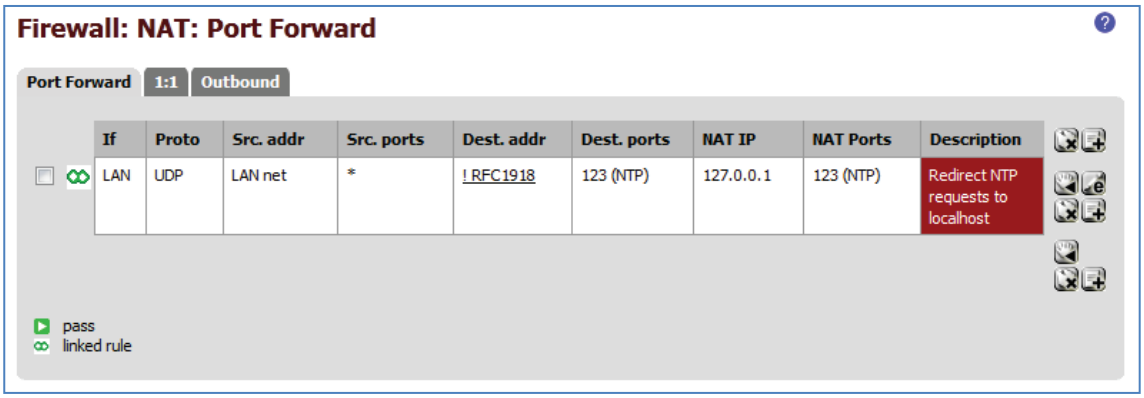

Figura 43. Configuración red NAT - Pfsense

3.6.1.9 ACTIVACIÓN DE ANTIVIRUS

Todos los sistemas por defecto vienen preconfigurados con el servicio de antivirus activo, mismo que puede ser activado o desactivado por interfaz gráfica o comando

```
]# service clamd restart
Stopping clamav:                                           [  OK  ]
Starting clamav: LibClamAV Warning: ************************************************************
LibClamAV Warning: ***  This version of the ClamAV engine is outdated.     ***
LibClamAV Warning: *** DON'T PANIC! Read http://www.clamav.net/support/faq ***
LibClamAV Warning: ************************************************************
```

3.6.1.10 ACTIVACIÓN DE FILTRADO DE CONTENIDOS

La temática de filtrado de contenido (Figura 44) en los sistemas OPENSOURCE sigue un mismo patrón, descarga de categorías de listas negras, carga de archivo y selección de categorías, ampliamente configurables en todos los sistemas, para que habilitar únicamente usuarios específicos a recursos WEB permitidos.

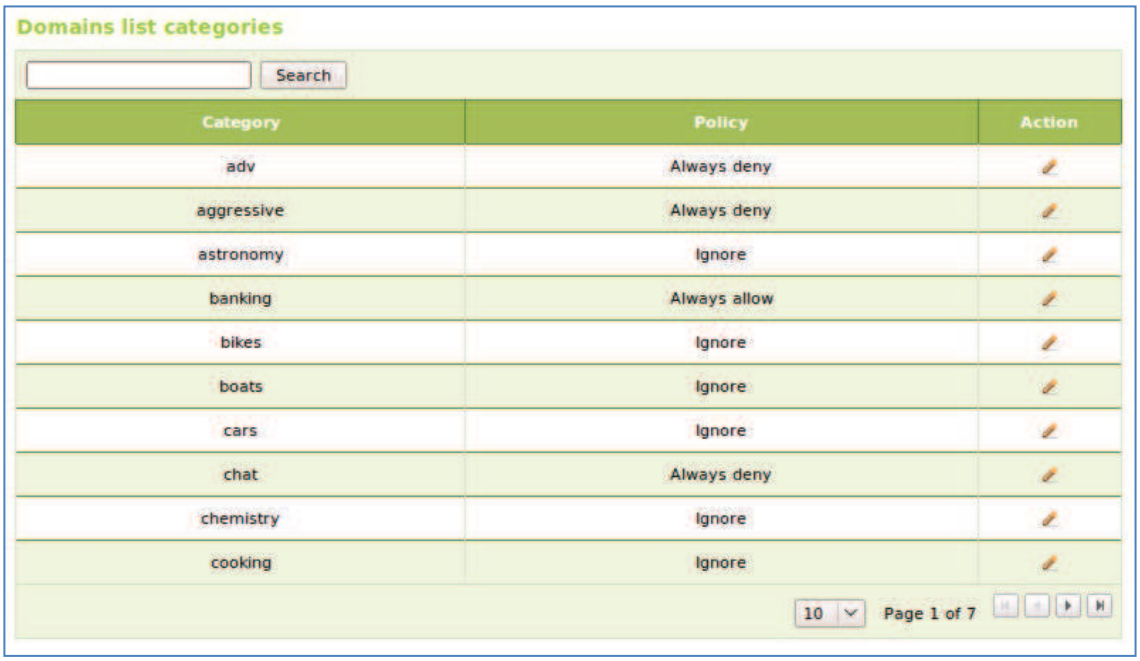

Figura 44. Configuración de filtrado de contenidos

3.6.1.11 OBTENCION DE RESULTADOS

Una vez conformado el ambiente experimental se establecieron los procedimientos y los tipos de ataques para la ejecución de las pruebas. Las pruebas fueron orientadas para determinar el desempeño, consumo de CPU, memoria, de los sistemas de firewall. El esquema de pruebas para calcular el consumo de recursos de CPU y Memoria del firewall, fue establecido en base algunas métricas: Nº de páginas Web solicitadas; tiempo de duración de la prueba y periodo de solicitud de páginas Web.

Las pruebas fueron efectuadas desde un equipo de la red local, el mismo que en base a la generación de una secuencia de comandos (algoritmo) de procesamiento en bach, generaba la solicitud de múltiples conexiones hacia páginas Web, esto en los siguientes escenarios enumerados en la Tabla1, conducidos por una serie de ataques que se detallan en la Tabla 4:

ESCENARIOS AVALUADOS
NAT+ Filtrado de contenido
NAT+ Filtrado de contenido+ Antivirus
NAT+ Filtrado de contenido+Antivirus+ IPS

Tabla 4. Escenarios evaluados, configurados en los sistemas de firewalls

ATAQUES	HERRAMIENTA
Escaneo de puertos	ZenMap, nmap
Sniffing	The dude
Denegación de Servicios	Loic

Tabla 5. Tipos de ataques y herramientas utilizadas

3.6.2 Algoritmo de simulación de solicitud de varias páginas Web

En un ambiente de pruebas se debe considerar como un limitante tecnológico real la disponibilidad de equipos informáticos desde los cuales se debería realizar las distintas solicitudes de sitios Web. Tampoco es aconsejable realizar pruebas de ataques en redes en una red en producción. Para este experimento, en las pruebas se debía realizar la petición de un número determinado de páginas Web, para lo cual se debería disponer de varios equipos que realicen diversas solicitudes y entre todos estos sumar las peticiones deseadas. Por ejemplo desde 10 computadores diferentes se debería realizar 10 consultas simultáneas, lo cual se torna oneroso ya que los usuarios deberían realizar las solicitudes al mismo tiempo y en una secuencia exacta, resultando una

práctica ineficiente, debido a la falta de sincronización entre ellos. Como alternativa de solución se implementó un archivo de procesamiento por lotes batch, que permita realizar múltiples peticiones de páginas Web desde un solo equipo simultáneamente y en periodos definidos.

Cuando se ejecuta este algoritmo, los comandos contenidos son ejecutados en grupo, de forma secuencial, permitiendo automatizar diversas tareas como la lectura del archivo de texto plano en el cual se encuentran ingresadas las diferentes direcciones (URLS) de las páginas Web a descargar. Posteriormente, después de 15 segundos de iniciada la ejecución se inicializa o cierra el navegador predeterminado, volviéndose a ejecutar la petición de las páginas que se encuentran en el archivo, todo esto encerrado en un lazo cíclico para n número de repeticiones. La cantidad de páginas y el periodo de tiempo son configurables. A continuación, la Figura 45 muestra el diagrama de flujo y el código del algoritmo.

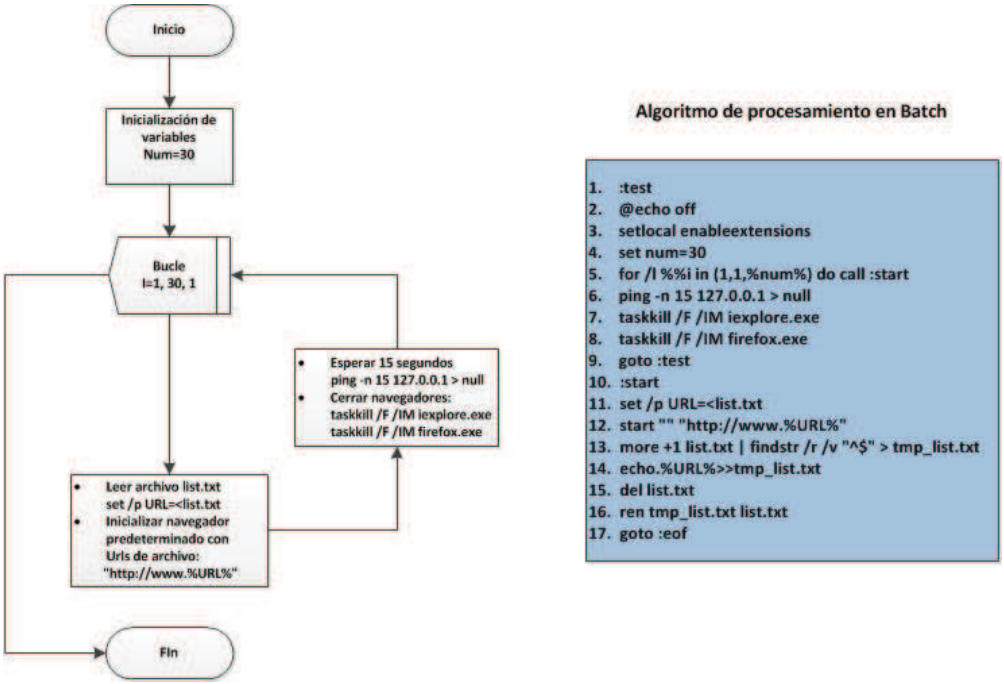

Figura 45. Algoritmo que simula la llamada de 100 páginas Web

CAPITULO IV

EVALUACIÓN DE RESULTADOS PLATAFORMAS SELECCIONADAS

4.1 INTRODUCCIÓN

Tal como se explicó en la Introducción, esta investigación, se enfocó en evaluar tres sistemas de firewall de código abierto en base al sistema propietario WATCHGUARD y repotenciar la plataforma de firewall basado en las características distintivas del sistema propietario. Por tanto, al comparar su comportamiento en términos de desempeño y efectividad, nos permitiría elegir aquella herramienta que reúna las mejores características y repotenciarla.

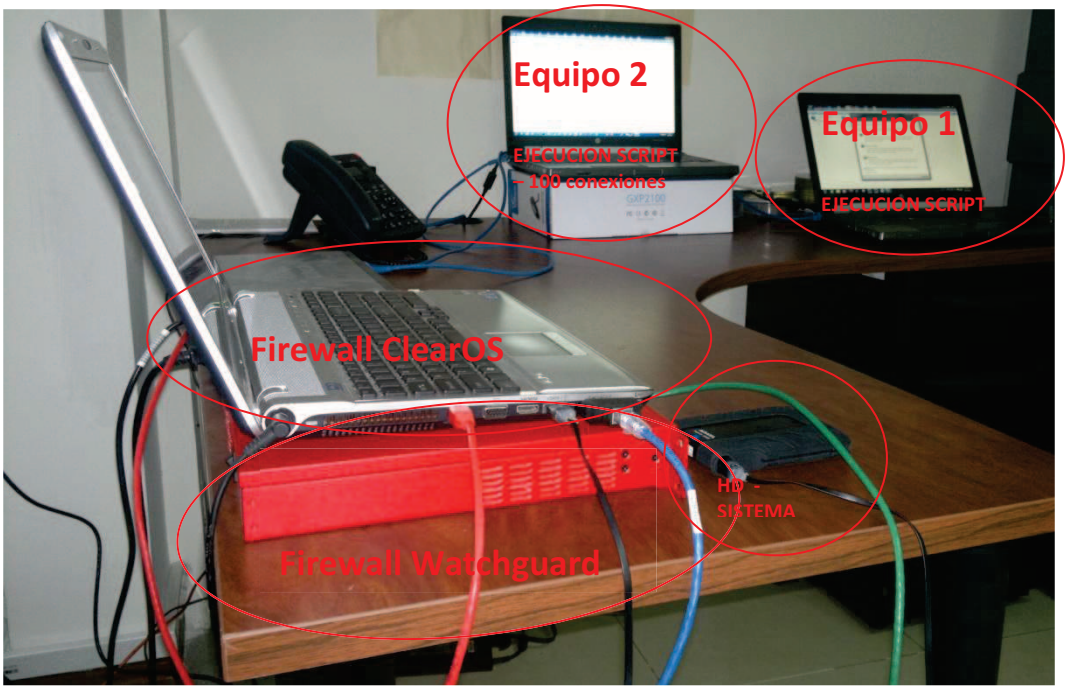

Figura 46. Topología de prueba

El procedimiento consistió en ejecutar scripts al interior del firewall por 60 segundos desde el equipo 1 de la topología de prueba (Figura 46). Al mismo tiempo el equipo 2 ejecuta el script de 100 conexiones http. Luego de 60 segundos en el equipo 3 se respalda y procesa los registros generados en el firewall. El procedimiento descrito estaba orientado para medir el consumo de CPU, memoria, y su rendimiento, en los cuatro sistemas de firewall, en los escenarios descritos en la Tabla 4 y durante los ataques establecidos en la Tabla 5. Los resultados se muestran en las Figuras 47 - 54.

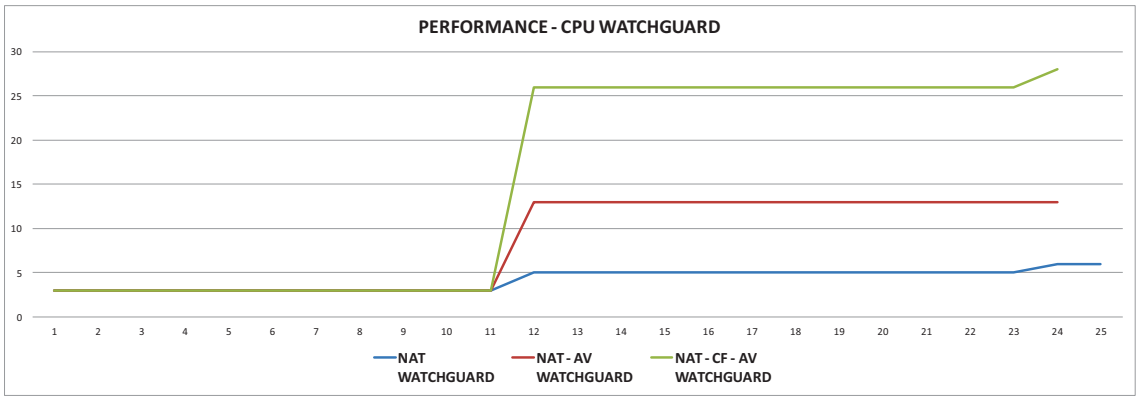

Figura 47. Uso de procesamiento por unidad de tiempo - WATCHGUARD

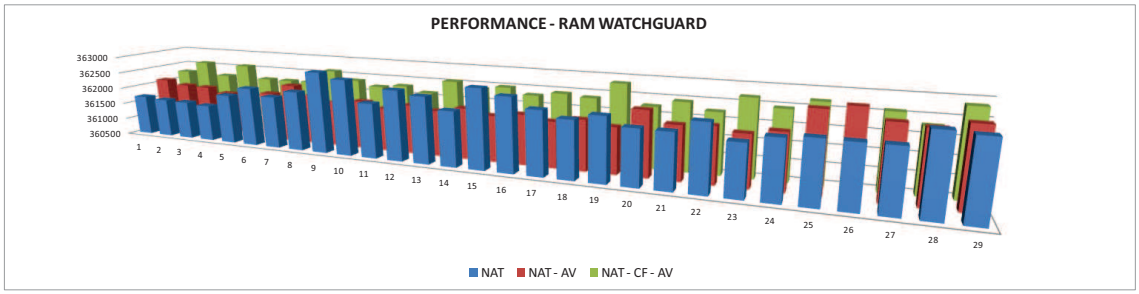

Figura 48. Uso de memoria por unidad de tiempo - WATCHGUARD

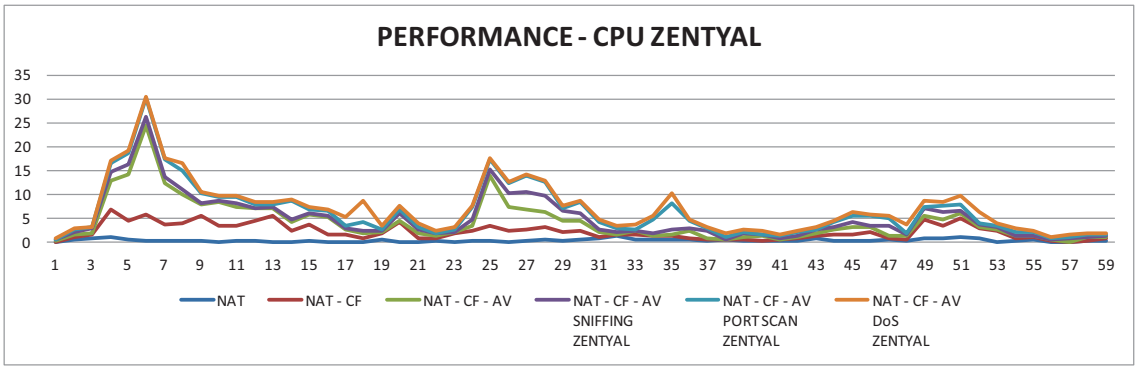

Figura 49. Tendencia del uso de procesamiento por unidad de tiempo - ZENTYAL

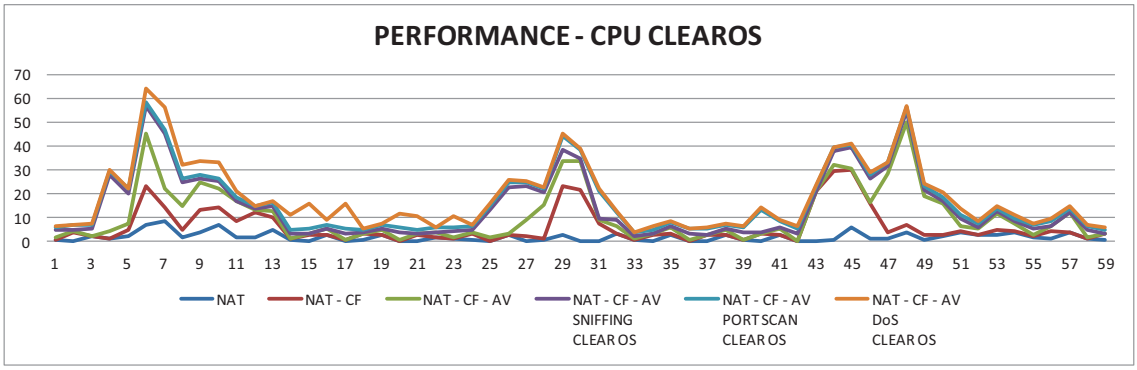

Figura 50. Tendencia del uso de procesamiento por unidad de tiempo - Clear OS

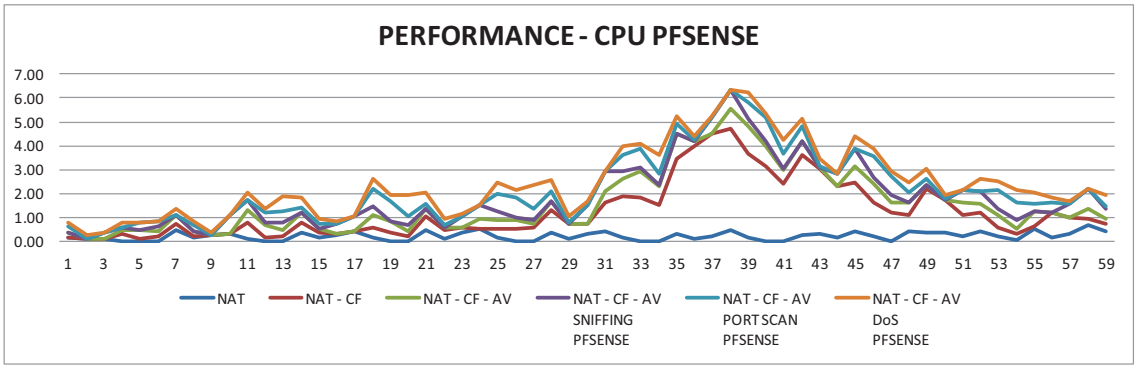

Figura 51. Tendencia del uso de procesamiento por unidad de tiempo – PFSense

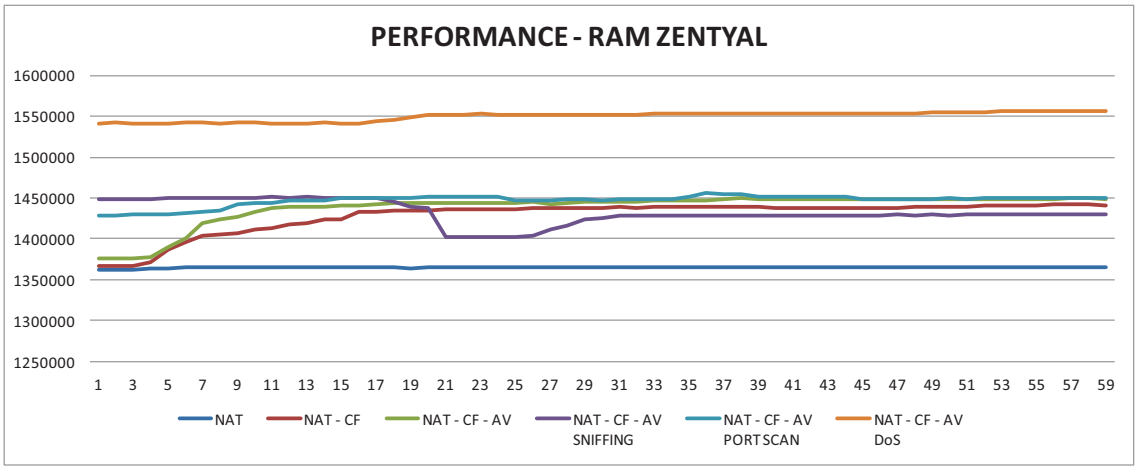

Figura 52. Uso de memoria por unidad de tiempo – Zentyal

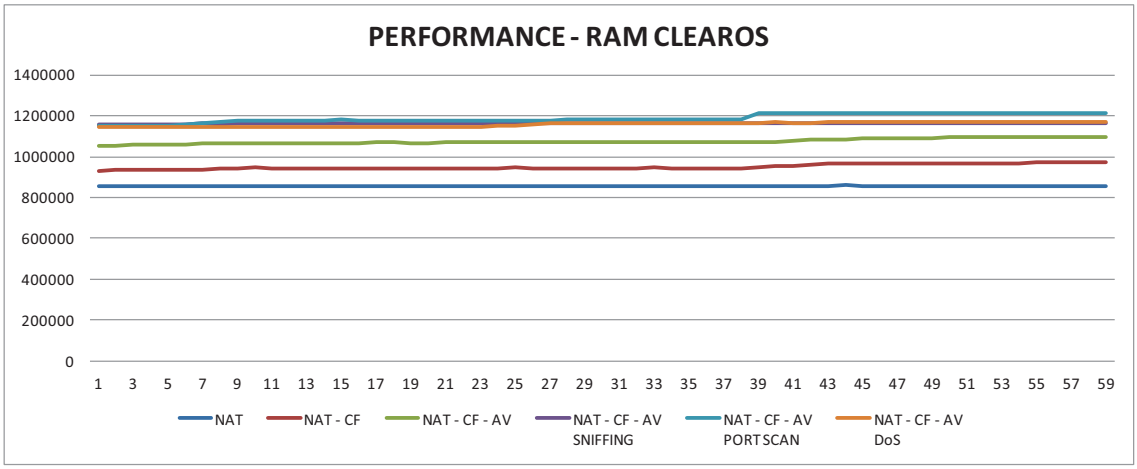

Figura 53. Uso de memoria por unidad de tiempo – ClearOS

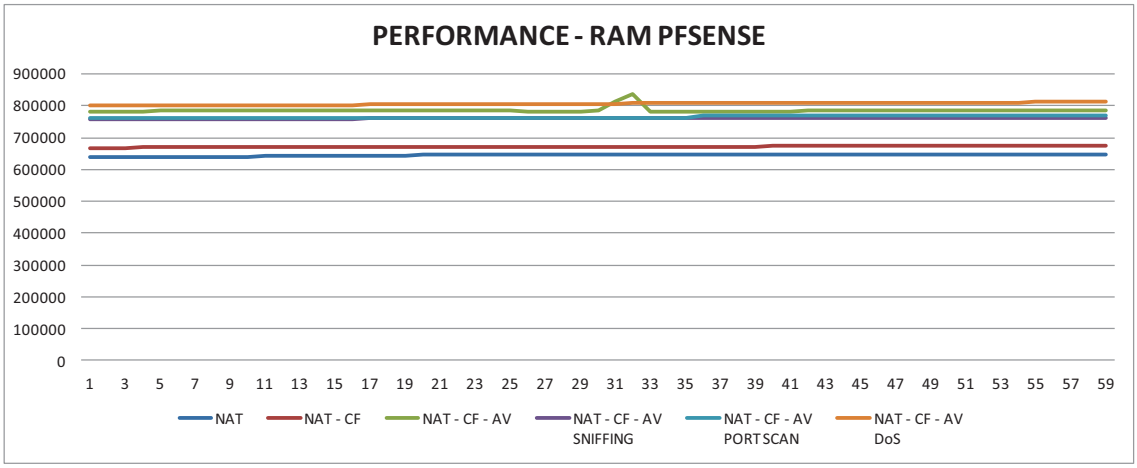

Figura 54. Uso de memoria por unidad de tiempo – Pfsense

Como se puede apreciar, el desempeño de cada uno de los firewalls analizados, es directamente proporcional al nivel de aplicabilidad (figuras 47-54) que se desee de los mismos. Al realizar una evaluación previa de los resultados en el entorno físico, se observa:

PFSense es el firewall que mejor optimiza sus recursos de hardware (Figura 51), pero su desempeño ante ataques de RED (figuras 56 - 58) y uso de aplicativos no estables lo hacen un sistema poco confiable a la hora de seleccionarlo.

Zentyal optimiza sus recursos de CPU (Figura 49), pero el uso de memoria fue superior (Figura 65) y su desempeño ante ataques de RED fue muy bajo (figuras 56 - 58).

ClearOS (SO RedHat) obtuvo resultados muy similares (Figura 50) a Zentyal (SO Debian) en el uso de los recursos de Hardware, sin embargo la optimización de memoria RAM (Figura 53), soporte técnico del sistema operativo y de aplicación y mejor desempeño ante actividades sospechosas de RED (figuras 56-58), permite seleccionar a esta herramienta para su posterior repotenciación en aspectos de seguridad.

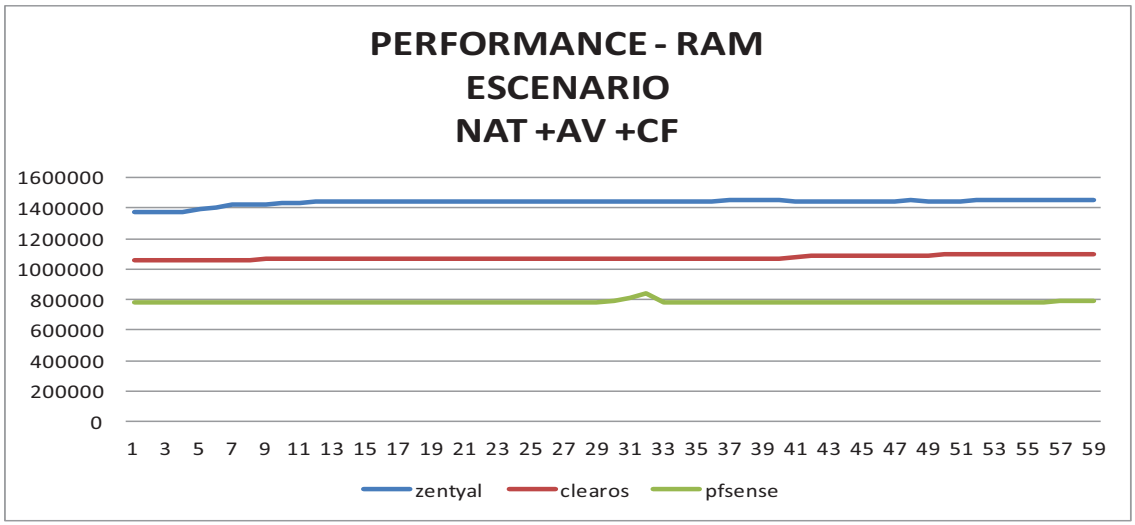

Figura 55. Desempeño de los tres sistemas de firewall - RAM

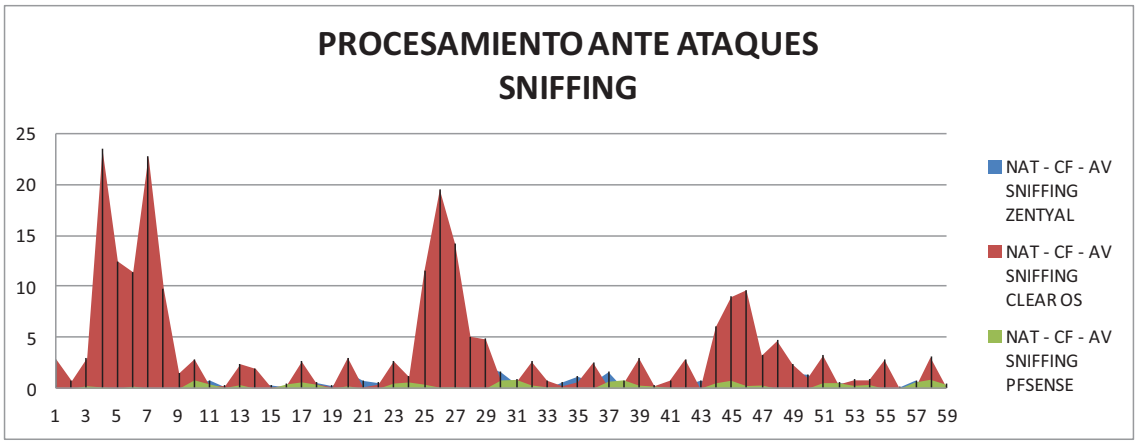

Figura 56. Desempeño de los tres sistemas de firewall ante ataques de RED - SNIFFING.

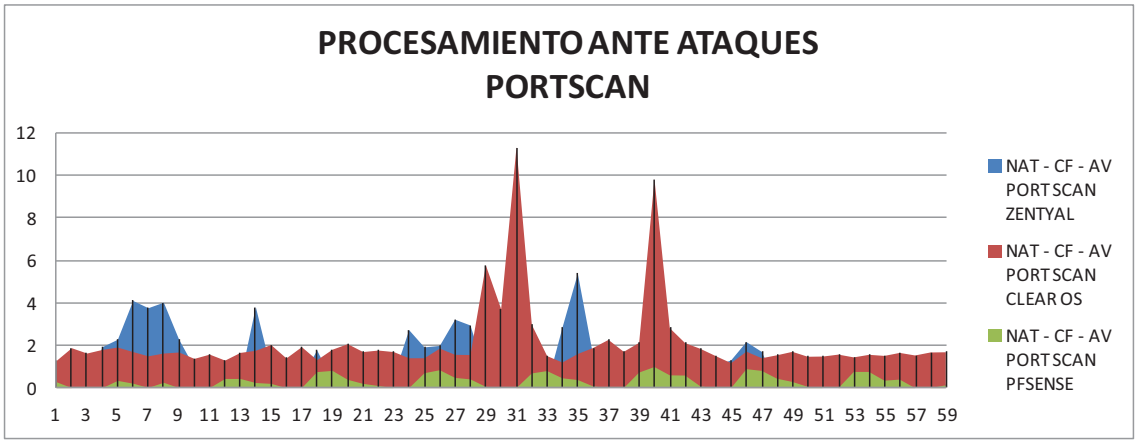

Figura 57. Desempeño de los tres sistemas de firewall ante ataques de RED - PORTSCAN.

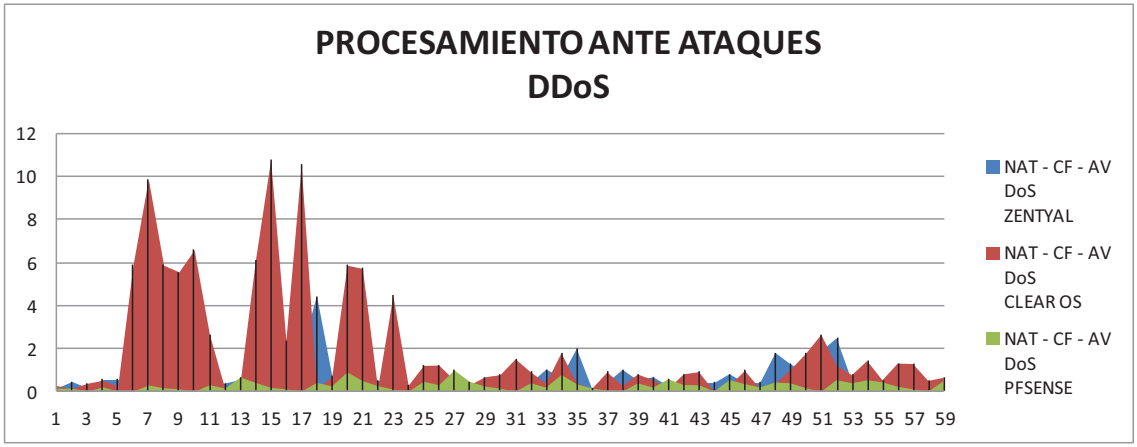

Figura 58. Desempeño de los tres sistemas de firewall ante ataques de RED - DDoS.

CAPITULO V

REPOTENCIACION FIREWALL SELECCIONADO - CLEAROS

Clear OS es una plataforma de servidor basada en Linux que se gestiona a través de una herramienta de administración en entorno Web. La mayoría del código fuente proviene de la distribución Red Hat Linux Enterprise. La programación se encuentra distribuida en capas y niveles y dispone de una arquitectura cliente-servidor. Así, la capa de presentación, contiene todas las páginas que ve el usuario, presenta e informa las actividades que se ejecutan; la capa de negocio, contiene los programas que se ejecutan el momento que el usuario precisa una tarea determinada; capa de datos, donde residen los datos.

Clear OS utiliza como herramienta de desarrollo a nivel de capa de presentación PHP para su diseño de entorno gráfico en ambiente Web y como servidor HTTP APACHE. Además en su capa de negocios utiliza como herramienta JAVA.

La repotenciación de la herramienta se basó en dos aspectos básicamente: implementación de IPS (PSAD), TCPdump, P0f, Nessus y el desarrollo del módulo PSAD en el GUI de ClearOS para complementar un entorno amigable con el usuario de la herramienta.

5.1 IMPLEMENTACIÓN DE IPS (PSAD), TCPDUMP, P0F, NESSUS

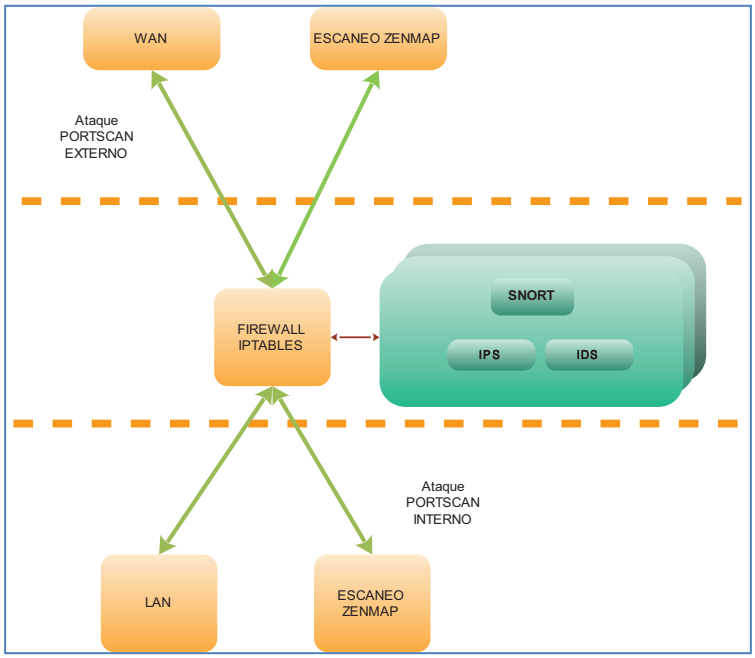

Figura 59. Desempeño de los tres sistemas de firewall ante ataques de RED - DDoS.

La Figura 59, en referencia al diseño de la topología de RED, indica que el firewall es un nodo de alto riesgo en la RED y que es muy susceptible a ataques internos como externos. Al instalar PSAD en el sistema base del firewall (Figura 60) se destaca que:

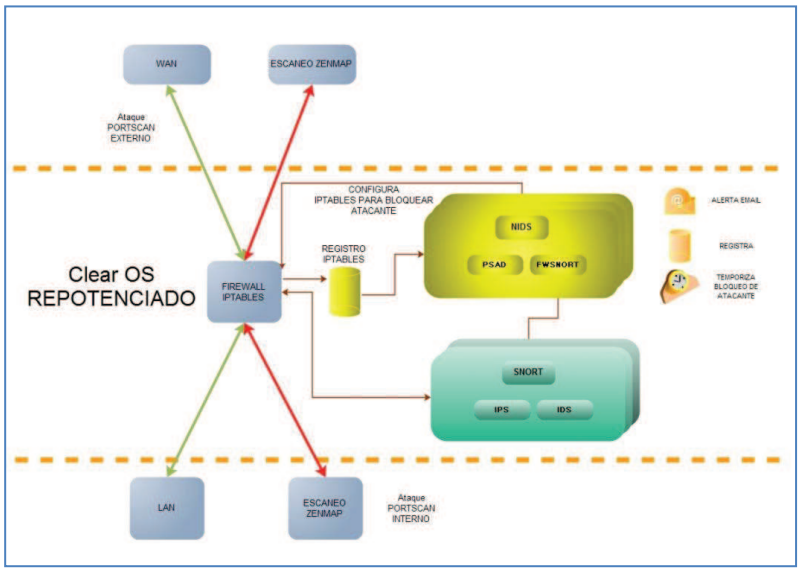

Figura 60. Repotenciación de CLEAROS.

Psad tiene una vinculación directa con snort e iptables, para realizar un bloqueo parametrizable. En la presente investigación se enlazó el estudio realizado "Alternative Engine to Detect and Block Port Scan Attacks using Virtual Network Environments", al método de detección, bloqueo y liberación de los equipos atacantes implementado por PSAD.

Esta repotenciación (ver Figura 60) consiste en verificar continuamente los contadores generados por PSAD, las direcciones IP consideradas maliciosas y su categoría (1-5), en concepto el motor alternativo filtra esta información y la procesa inmediatamente cuando un atacante sea categorizado (cat 3) por el número de paquetes que haya generado. Psad es una herramienta potente y actualizable, es por este motivo que filtra efectivamente el tráfico generado por programas de escaneo de puertos, sniffing y DDoS, tal como se pudo constatar en la presente investigación.

Previa la implementación de TCPdump (ver Figura 61) y Pf0 (ver Figura 62), se evaluó la inexistencia de conflictos de software y puertos en relación al sistema operativo,

mediante la instalación independiente en entorno virtual de prueba, observando en los registros del sistema no reporten daño en el sistema. Inmediatamente evaluados estos aspectos se procedió a implementar los sistemas ante mencionados en CLEAROS.

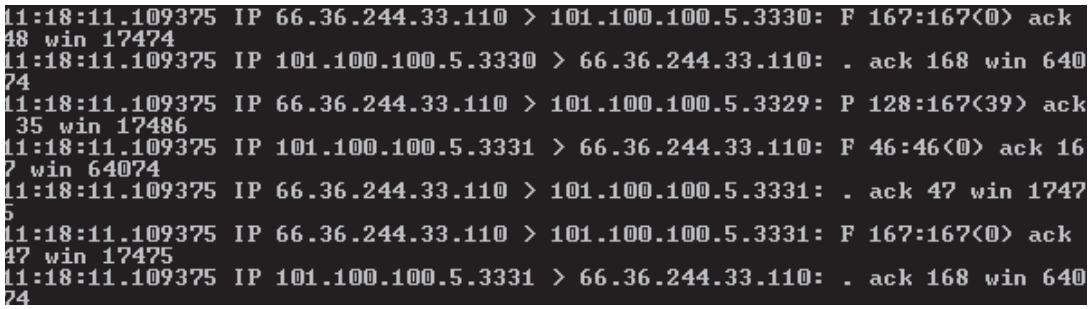

Figura 61. TCPDUMP en ejecución.

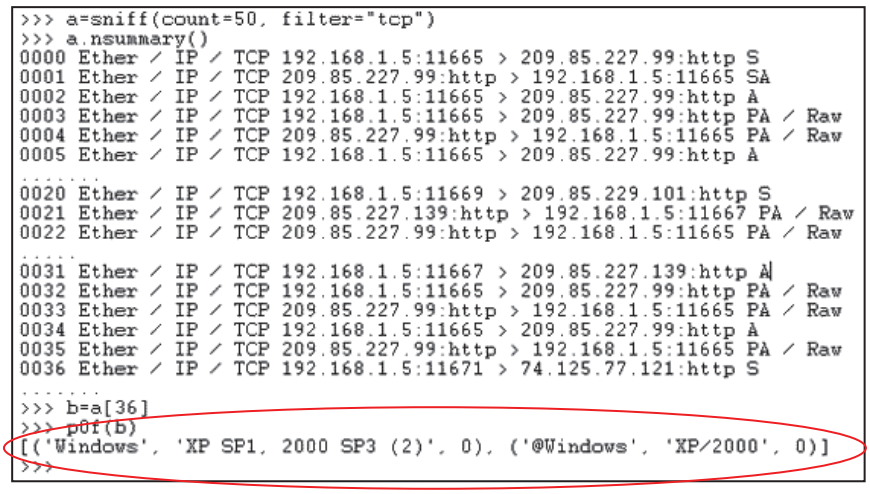

Figura 62. P0f en ejecución.

Nessus (ver Figura 14) al ser una herramienta que implementa una interfaz de usuario gráfica, habilita puertos de acceso y configuración de archivos, por este motivo que se obtuvo un paquete estable del autor de la herramienta y en prueba de laboratorio se certificó que tanto su instalación como funcionamiento no generara conflictos a la hora de interoperar con la interfaz gráfica de CLEAROS.

5.2 DESARROLLO DE MÓDULO PSAD

La Figura 63 muestra la GUI (Interfaz gráfica de usuario) de la distribución ClearOS en la cual se puede observar que en el menú de RED (Network) se ha incluido en módulo PSAD.

Figura 63. GUI Clear OS – Menú Network (PSAD)

Ingresando a la opción PSAD se presenta el resultado del desarrollo del módulo (Figura 64) parametrizado a las necesidades del proyecto.

Figura 64. GUI Módulo PSAD

5.2.1 ESTRUCTURA DE ARCHIVOS

ClearOS dispone de una estructura definida de archivos y directorios donde almacena todo su código e información relacionada con la aplicación.

Ubicación	Tipo	Propósito
/etc/clearos/basename	Archivo	Archivo de configuración (opcional)
/etc/clearos/basename.d	Directorio	Archivos Configlet (opcional)
/usr/clearos/apps/basename/	Directorio	Core de la aplicación (requerido)
/var/clearos/basename	Directorio	Archivos de estado (opcional)

Tabla 6. Estructura de directorios y archivos de ClearOs

El directorio principal donde residirá el nuevo módulo de la aplicación web es:

/usr/clearos/apps/Nombre_Aplicacion

Los módulos de ClearOs disponen de una estructura definida de sub-directorios los cuales contienen los archivos necesarios para que dichos módulos operen adecuadamente.

Directorio	Descripción
controllers	Capa de lógica de negocio
deploy	Contiene scripts de instalación, configuraciones predeterminadas
htdocs	Recursos web: Javascript, imagines
Info	Metadata
language	Traducciones
libraries	Capa de Datos
packaging	Información del paquete
tests	Pruebas de unidad
views	Capa de presentación

*Tabla 7.*Subdirectorios ClearOs

El desarrollo del módulo se lo realizo bajo el nombre de **PSAD** y de acuerdo a la definición de directorios que maneja ClearOs se lo colocó en la siguiente ubicación:

/usr/clearos/apps/psd

En base a la estructura de contenidos, librerías y código de ClearOs se crearon cuatro directorios en los cuales residirá la aplicación:

- controllers (Clases de lógica del negocio)
- view (Vistas o Capa de presentación)
- deploy (Archivos de configuración e información del módulo)
- language (Archivos de idioma)

5.2.2 CONTROLADORES

La aplicación se la accede a través de URL amigables las cuales son direcciones de fácil recordatorio para el visitante de un sitio web y representan clases del módulo. Al momento tenemos creadas 2 clases.

- Clase principal PSAD
- Clase controladora de vista Status

5.2.2.1 Clase Principal PSAD

Al controlador principal se accede ingresando la dirección https://direccion_ip:81/app/psad, el mismo que se encuentra en el directorio /usr/clearos/apps/psad/controllers/psad.php. De forma predeterminada este llama el método index ( ) (Figura 64), que a su vez hace un llamado a la vista status la cual muestra la sección NIDS e IPs bloqueadas.

```php
class PSAD extends ClearOS_Controller
{
    /**
     * Firewall server overview.
     *
     * @return view
     */

    function index()
    {
        // Load libraries
        //---------------

        $this->lang->load('psad');

        // Load views
        //-----------

        $views = array('psad/status');

        $this->page->view_forms($views, lang('psad_app_name'));
    }
}
```

*Figura 65.*Código clase controladora

5.2.2.2 Clase controladora de la vista STATUS

La vista solicitada dispone de su propio controlador, clase status, la cual se encuentra en el directorio /usr/clearos/apps/psad/controllers/status.php

Cabe mencionar que en primer lugar se usa la clase File (Figura 65), misma que utilizamos para lectura y escritura de archivos.

```php
///////////////////////////////////////////////////////////////////////////////
// D E P E N D E N C I E S
///////////////////////////////////////////////////////////////////////////////

use \clearos\apps\base\File as File;
```

Figura 66. Dependencia

En la Figura 66 se puede observar como el método principal realiza la lectura de un archivo de configuración, el cual es recibido como un arreglo y de esto se obtienen las variables nombradas administrador y sensibilidad. De la misma manera se obtiene en un arreglo las direcciones IPs (internet protocol) bloqueadas de un segundo archivo, el

mismo que es procesado para formar los botones de eliminación y textos. Finalmente llama a la vista para que se despliegue la información.

```
function index()
{
    // OBTENER ARCHIVO DE CONFIGURACION
    $test_file = new File("/usr/clearos/apps/psad/htdocs/testfile.txt");
    $data['contenido'] = $test_file->get_contents_as_array();

    foreach($data['contenido'] as $v){
        $item = explode(':',$v);
        if($item[0] == 'administrator')
            $data['administrator'] = trim($item[1]);
        else if($item[0] == 'sensibilidad')
            $data['sensibilidad'] = trim($item[1]);
    }

    // IPs Bloqueados
    $ip_file = new File("/usr/clearos/apps/psad/htdocs/bloqueados.txt");
    $ip_line = $ip_file->get_contents_as_array();
    $items = array();
    foreach($ip_line as $ip){
        $detail_buttons = button_set(
            array(
                anchor_delete('/app/psad/delete/'.$ip),
            )
        );

        $item['title'] = $ip;
        $item['action'] = '/app/dns/edit/';
        $item['anchors'] = $detail_buttons;
        $item['details'] = array($ip);
        $items[] = $item;
    }

    $data['items'] = $items;

    // Load views
    //-----------

    $this->page->view_form('psad/status', $data, lang('psad_status'));
}
```

*Figura 67.*Código función INDEX()

5.2.3 VISTAS

La Figura 67 nos visualiza como este repositorio presenta la información del módulo, que actualmente disponemos.

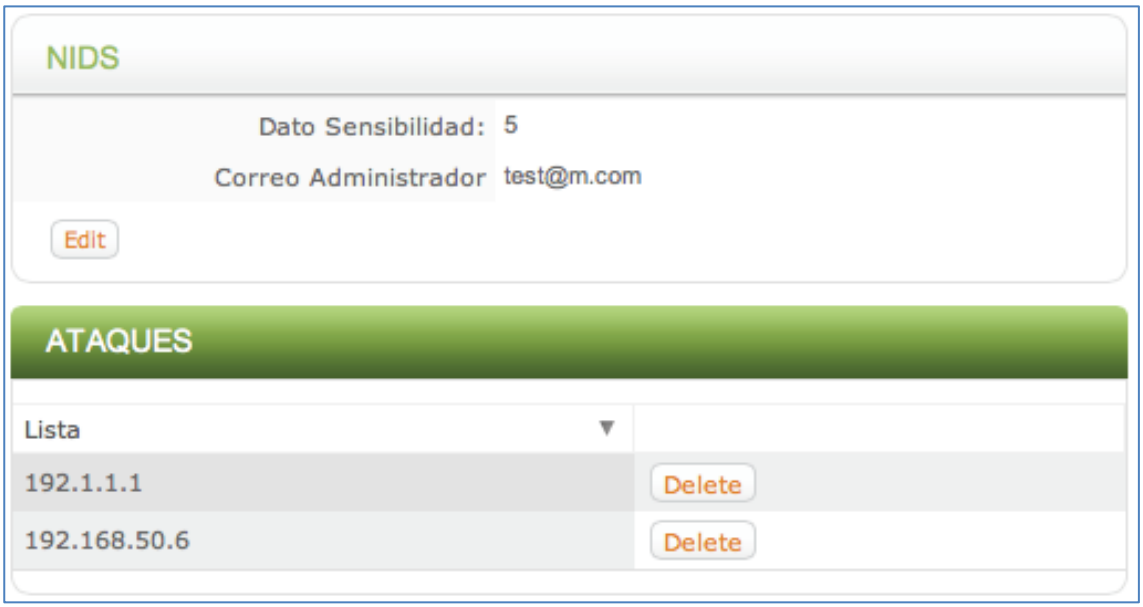

Figura 68. Vista principal NIDS

Es necesario que la vista se encuentre en el siguiente directorio:

/usr/clearos/apps/psad/views/status.php

5.2.3.1 Edición de configuración STATUS – Método edit

Dentro de la clase STATUS encontramos el método edit (Figura 68), el cual nos muestra el formulario de edición de la configuración de NIDS, y a su vez nos permite realizar la actualización.

```
function edit(){
    // OBTENER ARCHIVO DE CONFIGURACION
    $test_file = new File("/usr/clearos/apps/psad/htdocs/testfile.txt");
    $data['contenido'] = $test_file->get_contents_as_array();

    // Set validation rules
    //---------------------

    $this->form_validation->set_policy('administrador', 'psad/Validation',
    $this->form_validation->set_policy('sensibilidad', 'psad/Validation',
    $form_ok = $this->form_validation->run();

    if($this->input->post('submit') && $form_ok){
        $this->load->library('psad/Validation');

        $line = array();
        foreach($data['contenido'] as $v){
            $item = explode(' ',$v);
            if($item[0] == 'administrator')
                $line[] = 'administrator: '.$_POST['administrador'];
            else if($item[0] == 'sensibilidad')
                $line[] = 'sensibilidad: '.$_POST['sensibilidad'];
            else
                $line[] = $v;
        }
        $test_file->dump_contents_from_array($line);
        redirect('/psad/status');
    }
```

Figura 69. Código método edit

a. Validación

Para las validaciones usamos la clase Form_validation (Figura 69) la cual a su vez requiere que creemos en el directorio de librerías la clase validadora:

/usr/clearos/apps/psad/libraries/Validation.php

Esta clase solo en caso de que la validación sea incorrecta debe mostrar un mensaje de error (Figura 70). La aplicación se encarga de mostrar los mensajes de error respectivos.

```
class Validation
{
    ////////////////////////////////////////////////////////////////////////////////
    // M E T H O D S
    ////////////////////////////////////////////////////////////////////////////////

    /**
     * Dnsmasq constructor.
     */

    public function __construct()
    {
        //clearos_profile(__METHOD__, __LINE__);
    }

    public function validate_email($address){
         $valid = ( ! preg_match("/^([a-z0-9\+_\-]+)(\.[a-z0-9\+_\-]+)*@([a-z0-9\-]+
         if(!$valid)
             return 'Por favor ingrese un correo válido';
    }

    public function validate_sensibility($sensibilty){
         if(!is_numeric($sensibilty))
             return 'Por favor ingrese un número decimal';
    }
}
```

Figura 70. Código clase Validation

Figura 71. Presentación GUI de la validación

5.2.4 ELIMINACION IP BLOQUEADA – CLASE STATUS, MÉTODO DELETE ($ip)

Este método recibe como parámetro la IP y elimina la línea correspondiente del archivo con IPs bloqueadas (Figura 71); Y realiza una redirección al método main de la aplicación.(Figura 72)

```
function delete($ip)
{
    if($ip){
        // IPs Bloqueados
        $ip_file = new File("/usr/clearos/apps/psad/htdocs/bloqueados.txt");
        $ip_line = $ip_file->get_contents_as_array();

        $line = array();
        foreach($ip_line as $v){
            if($v != $ip)
                $line[] = $v;
        }

        $ip_file->dump_contents_from_array($line);
    }

    redirect('/psad/status');
}
```

Figura 72. Código método Delelte ($ip)

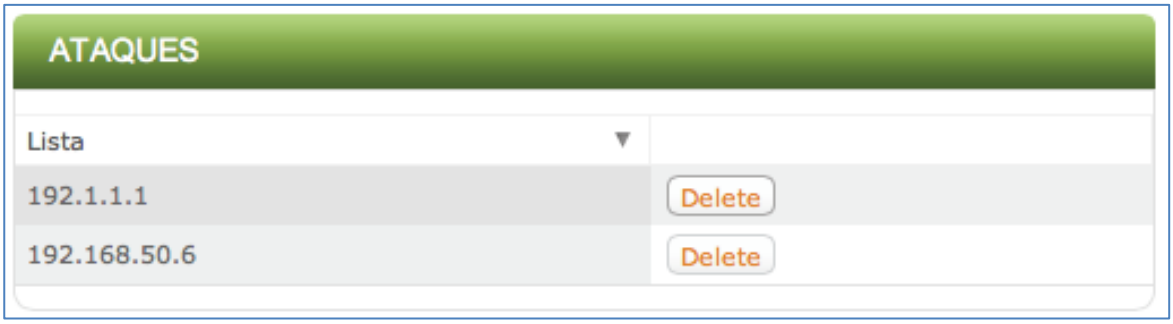

Figura 73. Presentación GUI

5.2.5 ARCHIVOS DE CONFIGURACIÓN E INFORMACIÓN DEL MÓDULO

Se encuentra en el directorio:

/usr/clearos/apps/psad/deploy/info.php

Este contiene el nombre del módulo, versión, autor entre otros datos. También contiene la categorización que se ve reflejada en el menú principal así como en el vertical.

```
$app['name'] = lang('psad_app_name');

$app['category'] = lang('base_category_network');

$app['subcategory'] = lang('base_subcategory_infrastructure');
```

5.2.5.1 Archivos de idioma

Se encuentra en el directorio:

/usr/clearos/apps/psad/language/en_US/psad_lang.php

Para acceder al nombre de una variable basta con llamar desde la vista con lang('nombre_input')

5.2.6 Servicios

Para poder integrar el módulo con un servicio es necesario crear el controlador server.php dentro del directorio controllers. Este contendrá una clase Server (Figura 73) la cual utiliza métodos de la clase padre Daemon. Métodos como por ejemplo iniciar, detener y conocer un estado de un servicio.

En el constructor se instancia al objeto con dos parámetros, el primero el nombre del servicio, como ejemplo mysqld, y el segundo el módulo que controla este servicio.

```
class Server extends Daemon
{
    /**
     * Server constructor.
     */

    function __construct()
    {
        parent::__construct('mysqld', 'psad');
    }
```

Figura 74. Código de clase Server - Constructor

Finalmente es necesario llamar a la vista Server en el método index (Figura 74) de la clase principal de la aplicación. De esta manera la aplicación automáticamente muestra las opciones de estado del servicio en la sección de información del módulo.

```
class PSAD extends ClearOS_Controller
{
    /**
     * Firewall server overview.
     *
     * @return view
     */

    function index()
    {
        // Load libraries
        //---------------

        $this->lang->load('psad');

        // Load views
        //-----------

        $views = array('psad/status','psad/server');

        $this->page->view_forms($views, lang('psad_app_name'));
    }
}
```

Figura 75. Código de clase PSAD – Método index()

5.3 EVALUACIÓN DE RESULTADOS – FIREWALL REPOTENCIADO CLEAROS

Las Figuras 75-78 muestran el performance de las herramientas Watchguard, ClearOS y ClearOS repotenciado bajo el mismo entorno de prueba, acentuando que el procesamiento de ClearOS repotenciado es superior al Base y a Watchguard bajo la misma carga, destacando como máximos valores de CPU intervalos de 25 a 35% porcentaje de uso. En referencia al uso de la memoria RAM se destaca que las plataformas manejan un rango de 1.1GB Watchguard - 1.3GB ClearOS

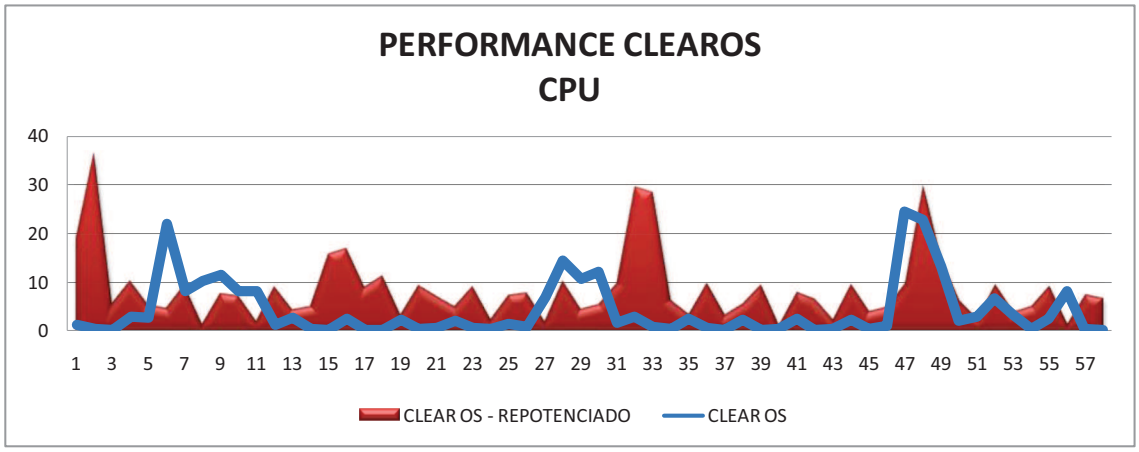

Figura 76. Uso de procesamiento por unidad de tiempo - Clear OS Repotenciado

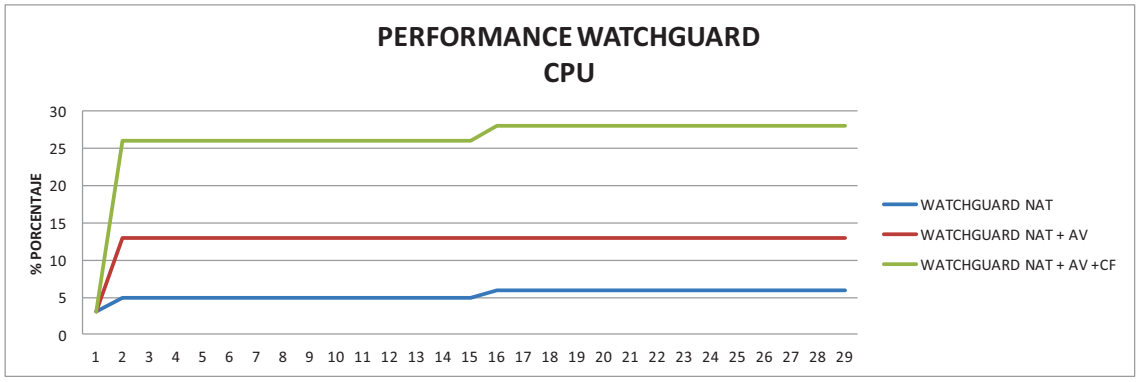

Figura 77. Uso de procesamiento por unidad de tiempo – Watchguard

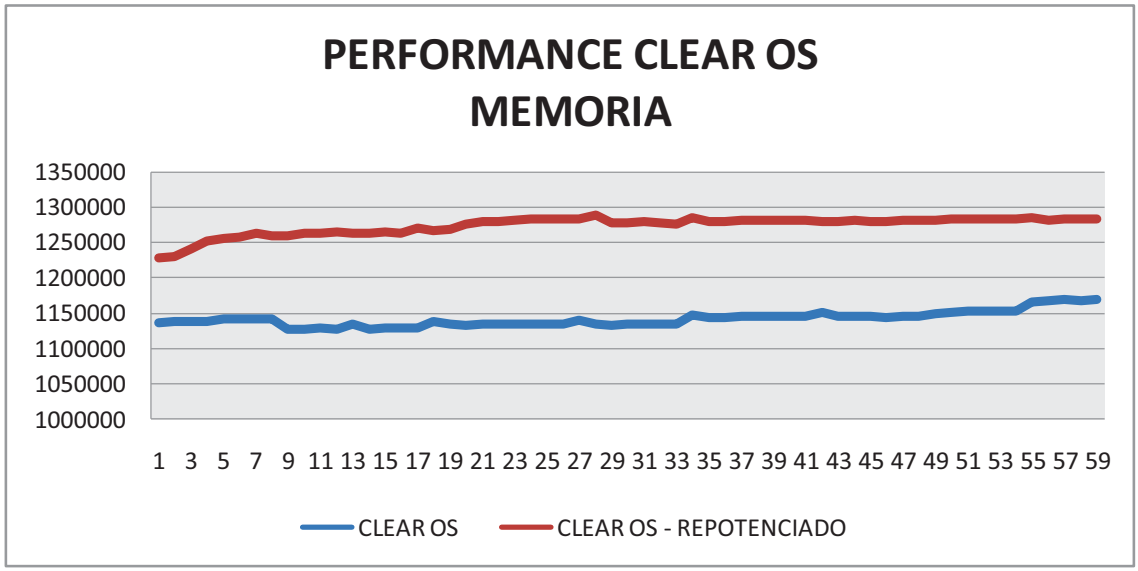

Figura 78. Uso de memoria por unidad de tiempo - Clear OS Repotenciado

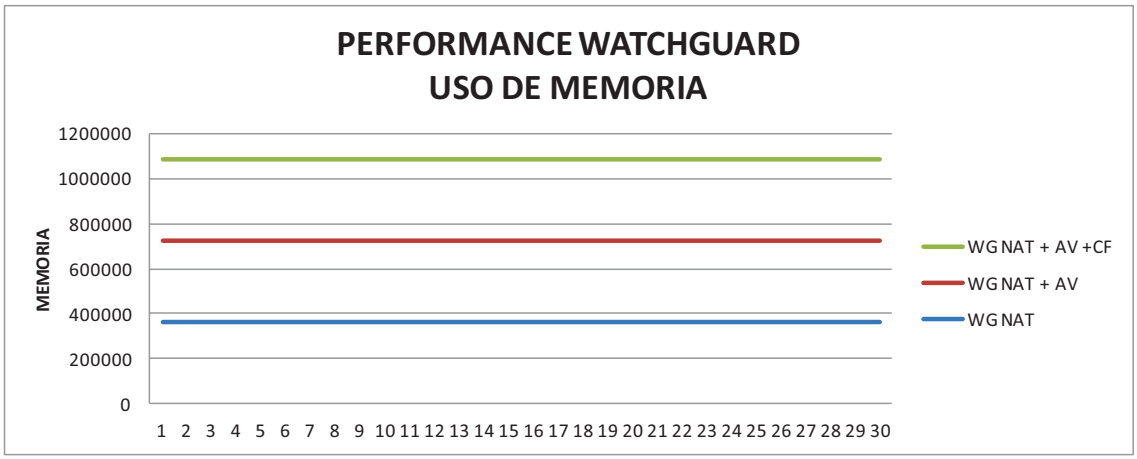

Figura 79. Uso de memoria por unidad de tiempo - Watchguard

En la Figura 79, se puede observar cómo se mantiene la continuidad del enlace permanente durante el intervalo de prueba hacia los firewalls desde un equipo atacante (de manera independiente), y una vez realizado el escaneo de puertos, en un tiempo controlado, la reacción del equipamiento Watchguard es casi inmediata (a los 8 segundos) seguida por el sistema repotenciado CLEAROS (a los 9 segundos) bloqueando inmediatamente la conexión entre el equipo atacante y el firewall por un tiempo descrito en la configuración de los mismos (10 segundos). De manera similar Watchguard y ClearOS repotenciado reaccionan ante un ataque con software sniffing (Figura 80) pero esta vez de manera más efectiva (5 y 7 segundos respectivamente.

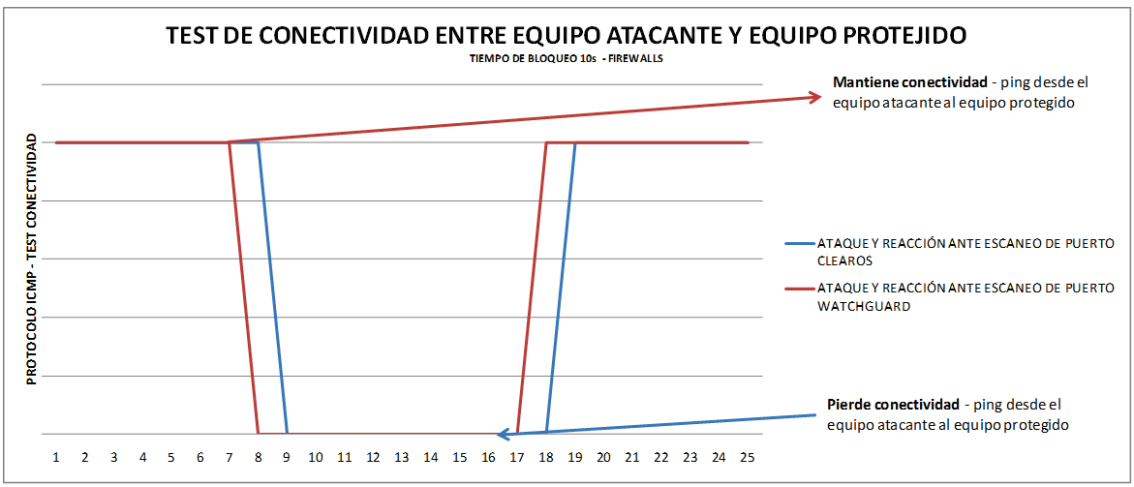

Figura 80. Test de conectividad entre atacante (portscan) y Firewalls

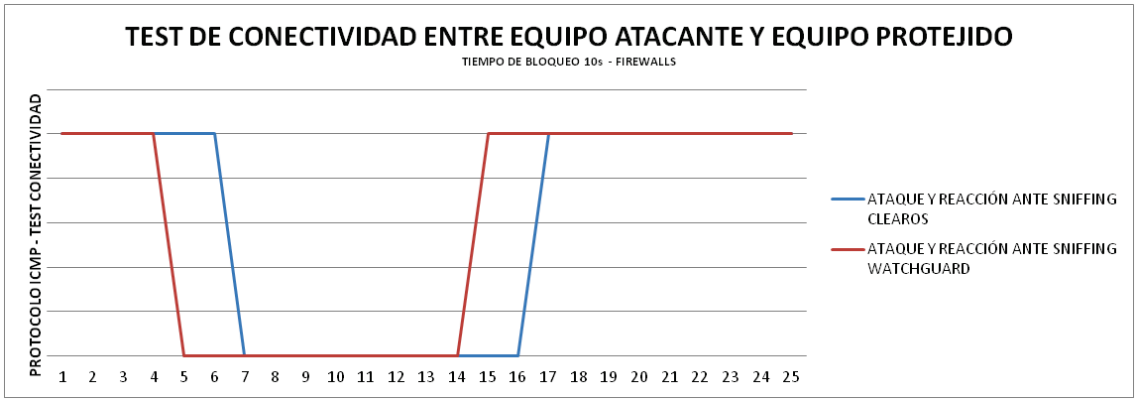

Figura 81. Test de conectividad entre atacante (sniffing) y Firewalls

La Figura 81, muestra la conectividad entre un equipo atacante y Watchguard sin que este tenga una reacción de bloqueo, ante el ataque de LOIC (1000 threads – ataque al protocolo http), aspecto que ClearOS repotenciado con PSAD no pasa por alto y bloquea al atacante a los 9 segundos por un periodo de tiempo paramentrizado en la herramienta.

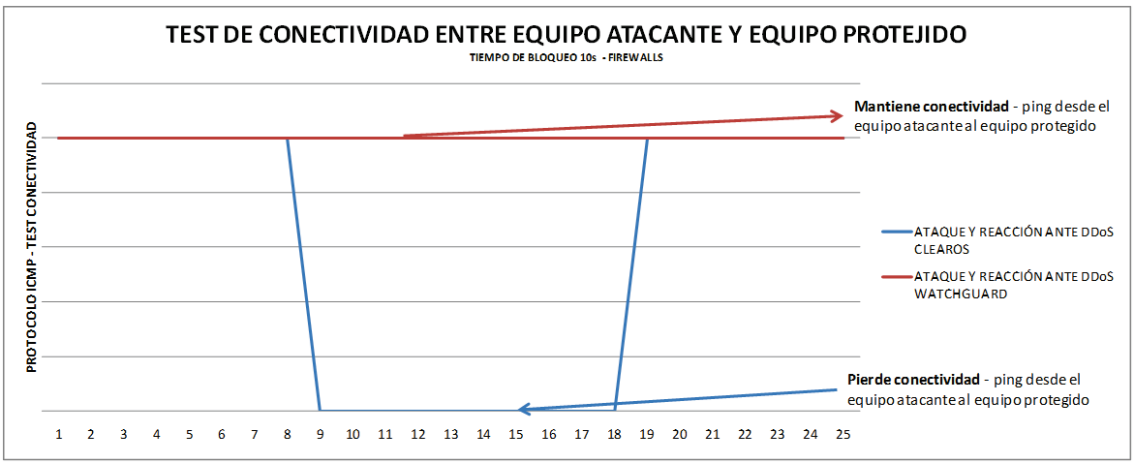

Figura 82. Test de conectividad entre atacante (DDoS) y Firewalls

CAPÍTULO VI

CONCLUSIONES Y RECOMEDACIONES

6.1 CONCLUSIONES

En el proyecto se demostró la falta de mecanismos de mitigación ante ataques orientados a la obtención de información relevante, por parte de las herramientas OPENSOURCE evaluadas, lo que incrementa el desafío de desarrolladores e investigadores, en el campo de la seguridad informática .

En la continua carrera entre fabricantes y desarrolladores de Firewalls OPENSOURCE, se evidencia una brecha muy marcada en aspectos de seguridad. La presente investigación coadyuvará en el desarrollo de las plataformas OPENSOURCE, haciéndolas más competitivas en aspectos de performance y confiabilidad.

El performance de todas las plataformas analizadas en el proyecto, en aspectos de uso de recursos de hardware (Memoria - CPU) es muy similar, bajo entornos reales. Estos datos pueden utilizarse para la selección y adquisición de herramientas de seguridad informática.

CLAROS como distribución especializada en seguridad bajo el sistema operativo RED HAT, demostró su amplia escalabilidad y soporte a la hora de mejorar sus prestaciones a diferencia de las herramientas no seleccionadas.

Ante el ataque DDoS generado por la herramienta LOIC, WATCHGUARD demostró una potencial vulnerabilidad, misma que prueba que las plataformas propietarias también se enfrentan a retos de continua mejora.

6.2 RECOMENDACIONES

Con la configuración correcta cualquier sistema de FIREWALLING basado en GNU LINUX es más seguro, debido a que la lógica de los desarrolladores de este tipo de herramientas se enfoca en aspectos de seguridad, es aquí donde lo determinante en la selección es el sistema base, y soporte técnico de la herramienta en caso de utilizar OPENSOURCE .

Haciendo una comparativa costo beneficio, valorizando a la información, es importante determinar si las empresas o entidades públicas orientan su data en un equipamiento propietario, mismo que cuenta con un grupo de desarrolladores que constantemente actualizan sus políticas y mantienen estándares de calidad y seguridad o por medio de software OPENSOURCE que continuamente se encuentra en evolución, que es altamente personalizable y mejorable (repotenciar) a la medida de las necesidades.

Sin adecuadas políticas de seguridad (normativas legales) y capacitación a los empleados mal se podría hablar de seguridad incluyente, misma que obliga a todas las persona a responsabilizarse de la información, ya que no depende únicamente de un equipamiento sino de un adecuado manejo de la información y el compromiso de los mismos por resguardarla.

Bibliografía

Pfsense. (2 de 8 de 2012). Obtenido de http://www.pfsense.org/

Al-Shaer, E. H. (23 de Octubre de 2005). Conflict classification and analysis of distributed firewall policies. *IEEE*, págs. 2069–2084.

B. Hickman, D. N. (Abril de 2003). Benchmarking methodology for firewall performance. *RFC 3511*.

Beale, J. a. (2004). Snort 2.1 Intrusion Detection. *Syngress Media*.

Cabezas, L. (2004). *Manual Imprescindible de PHP5.* Madrid: Ediciones ANAYA.

Cobo, A. y Gómez, P. (2005). *PHP y MySQL: Tecnologías para el desarrollo de aplicaciones web.* España: Ediciones Díaz de Santos.

Date S., R. M. (2001). *Introducción a los sistemas de bases de datos.* Mexico: ALHAMBRA.

DragonJAR. (03 de Agosto de 2012). Firewalls - Diseño y Panorámica Actual. págs. http://www.dragonjar.org/firewalls-diseo-panormica-actual.xhtml.

Freed, N. (Octubre de 2000). Behavior of and Requirements for Internet Firewalls. *IETF*.

G. Misherghi, L. Y. (Diciembre de 2008). A General Framework for Benchmarking Firewall. *IEEE*, págs. 227-238.

H. Hu, G. A. (Junio de 2012). Detecting and resolving firewall policy anomalies. *IEEE*, págs. 318–331.

Hongxin Hu, G.-J. A. (2012). Detecting and Resolving Firewall Policy Anomalies. *IEEE*.

Hulst, J. Z. (2012). Firewall fingerprinting. *INFOCOM*, págs. 1728 – 1736.

JeeHyun Hwang, T. X. (2012). Performance Modeling and Analysis of Network. *IEEE*.

Khaled Salah, K. E. (Enero de 2012). Performance Modeling and Analysis of Network. *IEEE*.

Kroenke, D. (2003.). *Procesamiento de bases de datos: fundamentos, diseño e implementación.* España: Pearson Educación.

Lau, M. L. (2000). Firewall security: policies, testing and performance evaluation. *IEEE International*, págs. 116–121.

Macromedia, Inc. (2002.). *Utilización de Dreamweaver MX.* San Francisco: Macromedia, Inc.

Newman, D. (Agosto de 1999). Benchmarking Terminology for Firewall Performance. *IETF RFC 2647*.

OS, C. (2012). *Clear OS.* Obtenido de http://www.clearfoundation.com/

Peña, O. (2 de agosto de 2012). Las entidades públicas sufren unos 5.400 ataques al año. *ABC Hoy Tecnología*.

Psad. (2012). *Psad*. Obtenido de http://www.cipherdyne.org/psad/

Rash, M. (2007). Attack Detection and Response with IPtables, psad, and fwsnort. *Linux Firewalls*, págs. ISBN: 10.1-59327-141-7.

S. Patton, D. D. (2000). Open Source versus Commercial Firewalls: Functional. *Proceedings of the Conference on Local Computer Networks*, págs. 223-224.

Sanchez, M. P. (s.f.). *PHP.*

Schuba, C. L. (1997). On the Modeling, Design, and Implementation of Firewall Technology. *Doctoral dissertation, Purdue University*.

Sheth, C. a. (2011). Performance Evaluation and Comparative Analysis of Network Firewalls. *ICDeCom*.

Stallings, W. (2009). *Comunicaciones y Redes de Computadoras.* 6ta Edición: 2009.

Tecnología, A. H. (2012). Las entidades públicas sufren unos 5.400 ataques al año. *ABC Hoy Tecnología*, http://www.hoytecnologia.com/noticias/entidades-publicas-sufren-unos/173132.

Walter Fuertes, P. Z. (30 de Noviembre de 2011). Alternative Engine to Detect and Block Port Scan Attacks using Virtual Network Environments. *IJCSNS-International Journal of Computer Science and Network Security*.

Walter Fuertes, P. Z. (2012). *Repotenciación de un firewall de Código abierto basado en una Evaluación Cuantitativa.* Sogamoso, Colombia: III Encuentro Internacional y VII Nacional de Ingeniería de Sistemas EIISI 2012 - Universidad Pedagógica.

Whitten J., Bentley, L. y Barlow, V. (2004). *Análisis y diseño de sistemas de información. Tercera edición.* México: McGraw-Hill.

Xie, J. Z. (2011). Researh on Network Intrusion Prevention System Based on Snort. *IEEE*, 1133-1136.

Xpress Hosting. (20 de 08 de 2008). *http://www.xpress.com.mx/glosario.php*.

Yan, W., Zhang, Z., & Ansari, N. &. (2008). Revealing packed malware. *IEEE*, págs. 65-69.

Ying-Dar Lin, S.-T. Y.-Y. (2010). Integrating and Benchmarking Security Gateway with. *IEEE*.

Yongxin, Y. (2011). The comparative study on network firewalls performance. *IEEE*, págs. 427-430.

Zentyal. (2012). *Zentyal.* Obtenido de http://www.zentyal.com/

Zitting, P. E. (2001). An expert system for analyzing firewall rules. *Helsinki University of Technology*.

FECHA DE ENTREGA DE LA TESIS

El presente documento fue entregado en la Dirección de Postgrado, reposando en la Escuela Politécnica del Ejército desde:

Sangolquí, 20 de Mayo del 2013

Ing. Marco Polo Sánchez Aguayo

Ing. Patrício Xavier Zambrano Rodríguez

AUTORES

Ing. Rodrigo Silva

Coordinador MRIC