



Seguridad Informática
Técnicas de defensa comunes bajo variantes del
sistema operativo Unix

Autor: Juan Pablo Sarubbi

Director: Javier Blanqué

Universidad Nacional de Luján

Int. Ruta 5 y 7

6700 Luján, Buenos Aires

República Argentina

Año 2008

Seguridad Informática
Técnicas de Defensa: Mecanismos comunes bajo variantes del
sistema operativo Unix

Juan Pablo Sarubbi
Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
pablosarubbi@gmail.com

*“El Arte de la Guerra nos enseña
que no debemos depender de la posibilidad de que el enemigo no venga,
sino que debemos estar siempre listos a recibirlo.
No debemos depender de la posibilidad de que el enemigo no nos ataque,
sino del hecho de que logramos que nuestra posición sea inatacable.”*

El Arte de la Guerra, Sun Tzu

Resumen

El presente trabajo intenta dar un panorama acerca de las técnicas más comunes de recolección de información, detección de vulnerabilidades y defensa de variantes del sistema operativo *Unix* propietarias como *AIX* de *IBM*, *HP-UX* de *Hewlett Packard* y *Solaris* de *Sun*, diversas distribuciones de *Linux*, como *Debian*, *Mandriva*, *SuSE Linux*, *Ubuntu*, *Red Hat*; y variantes de *BSD*, como *OpenBSD*, *FreeBSD* y *NetBSD*. A su vez se presenta una introducción a los conceptos básicos de seguridad de la información y procedimientos, comentarios y conceptos generales, que sirven para la vida profesional de un usuario intensivo de sistemas de información computarizados. Dividido en seis capítulos, el lector encontrará:

Una primer parte introductoria donde se describen conceptos básicos, definiciones y modelos de seguridad, mecanismos de prevención y detección, ejemplos concretos, y recomendaciones básicas para tener en cuenta.

El capítulo 2 ofrece un paneo general de seguridad a nivel del servidor, verificación de archivos logs, conexiones activas y puertos abiertos.

En el capítulo 3 se desarrolla el tema de criptografía como seguridad en los datos: historia, funciones hash, criptografía simétrica, criptografía asimétrica, firma digital y herramientas de implementación como PGP y GnuPG.

El cuarto capítulo se sumerge en los sistemas de monitoreo, detección de amenazas y reacción, como técnicas anti-sniffers, sistemas de detección de intrusos, anti-escaneo de puertos, sistemas de prevención y aplicaciones que se encargan de dichas tareas como *Portsentry* y *Snort*.

En el capítulo 5 se define y describen los distintos modelos de firewalls, estrategias de diseño, plataformas, características e implementación de los dos sistemas más conocidos como son *Iptables* y *Packet Filter*.

El último capítulo analiza la seguridad Wi-Fi mediante pruebas de laboratorio con diferentes métodos de cifrado como son WEP y WPA. También se detallan consideraciones generales a tener en cuenta durante la elección de una plataforma segura mediante el uso de esta tecnología.

Palabras Claves:

Seguridad Unix, Seguridad de la Información, Hacking

Agradecimientos

Agradezco profundamente a toda la gente que me rodeó durante estos años; a mi familia, mis amigos y a Sonia, que entendieron mi falta de tiempo por dedicación al estudio; a mis amigos y compañeros de universidad, charlas, café, almuerzos y cenas, por compartir conmigo gran parte de su tiempo y sus conocimientos a lo largo de estos años de carrera; y a mis profesores, especialmente a Javier por brindarme su tiempo, conocimientos, visión y dedicación en este proyecto.

A todos ustedes, Gracias.

Índice

1.Introducción y conceptos previos.....	Pág. 1
1.1.Historia.....	Pág. 1
1.2.Justificación y objetivos.....	Pág. 3
1.3.El primer gran incidente de seguridad informática.....	Pág. 3
1.4.¿Qué es la seguridad?.....	Pág. 4
1.5.Modelos de seguridad	Pág. 5
1.5.1Seguridad por oscuridad.....	Pág. 5
1.5.2Seguridad del perímetro.....	Pág. 6
1.5.3Seguridad en profundidad.....	Pág. 6
1.6.¿Qué asegurar?.....	Pág. 7
1.7.Mecanismos de seguridad.....	Pág. 9
1.7.1Enumeración de mecanismos de prevención y detección.....	Pág. 10
1.8.Ejemplos básicos de ataques comunes.....	Pág. 11
1.8.1Ataque externo contra una BD corporativa	Pág. 11
1.8.2Ataque interno contra un Server Administrativo.....	Pág. 15
1.9.Recomendaciones Básicas.....	Pág. 16
1.10.Conclusión.....	Pág. 17
2.Seguridad a nivel del Servidor.....	Pág. 18
2.1.Verificación de accesos – Syslog.....	Pág. 18
2.1.1Ejemplo de acceso no válido logueado en el auth.log:.....	Pág. 18
2.1.2Sugerencias.....	Pág. 18
2.1.3Alternativas - Syslog-ng [syslogNg]:.....	Pág. 19
2.1.4Extensión de características de edición Open Source con Python.....	Pág. 20
2.1.4.1Soporte de Buffering de mensajes en disco.....	Pág. 20
2.1.4.2Acceso directo a Base de datos	Pág. 21
2.2. Chequeo de tráfico en la red.....	Pág. 21
2.2.1Netstat.....	Pág. 21
2.2.1.1Consulta de conexiones, puertos activos y pasivos.....	Pág. 22
2.2.1.2Consulta de Tablas de enrutamiento.....	Pág. 22
2.2.1.3Consulta de estadísticas de una interfaz.....	Pág. 22
2.2.2Ntop.....	Pág. 24
2.2.3Nessus.....	Pág. 25
2.2.3.1Historia.....	Pág. 25

2.2.3.2Estructura.....	Pág. 25
2.2.3.3Instalación.....	Pág. 25
2.2.3.4Funcionamiento.....	Pág. 26
2.2.4Nagios.....	Pág. 28
2.2.4.1Instalación.....	Pág. 28
2.2.4.2Configuración.....	Pág. 29
2.2.4.3Plugins.....	Pág. 33
2.2.4.4Ejecución:.....	Pág. 35
2.3.Conclusión.....	Pág. 36
3.Criptografía.....	Pág. 37
3.1.Historia.....	Pág. 38
3.2.Ejemplo del método de Ríel, o de Rail Fence.....	Pág. 40
3.3.Esteganografía.....	Pág. 41
3.4.Funciones Hash	Pág. 42
3.4.1MD5.....	Pág. 43
3.4.2SHA.....	Pág. 44
3.5.Criptografía Simétrica.....	Pág. 44
3.5.1Ejemplo de sistema simétrico.....	Pág. 45
3.5.2DES.....	Pág. 45
3.5.3AES (o algoritmo de Rijndael).....	Pág. 46
3.6.Firma Digital.....	Pág. 46
3.7.Criptografía Asimétrica.....	Pág. 47
3.7.1Funcionamiento.....	Pág. 47
3.7.2RSA.....	Pág. 47
3.8.PGP.....	Pág. 48
3.8.1Historia.....	Pág. 49
3.8.2Encriptación.....	Pág. 50
3.8.3Firma Digital.....	Pág. 51
3.8.4Firma con Encriptación:.....	Pág. 51
3.9.GnuPG.....	Pág. 52
3.10.Conclusión.....	Pág. 53
4.Sistemas de monitoreo.....	Pág. 54
4.1.Técnicas de Detección de Sniffers	Pág. 54
4.1.1Qué es y cómo funciona un Sniffer.....	Pág. 54
4.1.2Técnicas locales de detección:.....	Pág. 55

4.1.3Técnicas de detección remota:	Pág. 55
4.1.3.1Técnicas dependientes del sistema operativo	Pág. 56
4.1.3.2Técnicas no dependientes del sistema operativo	Pág. 56
4.1.3.2.1Test de DNS	Pág. 56
4.1.3.2.2Test de latencia	Pág. 56
4.1.3.2.3Test ARP	Pág. 57
4.1.4La lucha continua	Pág. 57
4.2.Sistema de Detección de intrusos	Pág. 57
4.3.Portsentry	Pág. 58
4.3.1Funcionamiento típico:	Pág. 59
4.3.2Configuración de Portsentry	Pág. 60
4.3.2.1 Modo Clásico o simple:	Pág. 60
4.3.2.2Modo Avanzado o stealth:	Pág. 60
4.3.3Ejemplo de configuración:	Pág. 60
4.3.4Resultados del ataque a Portsentry	Pág. 62
4.4.Snort	Pág. 62
4.4.1Estructura de las Reglas	Pág. 63
4.4.2Componentes de Snort	Pág. 64
4.4.2.1Decodificador de paquetes	Pág. 64
4.4.2.2Preprocesador	Pág. 65
4.4.2.3Motor de detección	Pág. 65
4.4.2.4Sistema de alertas y logueo	Pág. 65
4.4.2.5Módulo de salida	Pág. 66
4.4.2.6Resumen	Pág. 66
4.4.3Protección del IDS	Pág. 66
4.4.3.1Snort en una interfaz Stealth	Pág. 67
4.4.3.2Snort en una interfaz sin IP asignada	Pág. 67
4.5.Conclusión	Pág. 67
5.Firewall	Pág. 68
5.1.Tipos de firewall	Pág. 68
5.1.1Filtrado de paquetes, Screening Router o de capa de red:	Pág. 68
5.1.2Proxy-Gateway o de capa aplicacion:	Pág. 69
5.1.3Dual Homed Host	Pág. 70
5.1.4Screened Host	Pág. 71
5.1.5Screened Subnet	Pág. 72
5.1.6Firewall Personal	Pág. 73

5.2.Estrategias en diseños de firewalls.....	Pág. 74
5.3.Netfilter/Iptables.....	Pág. 74
5.3.1Historia.....	Pág. 74
5.3.2Operación.....	Pág. 74
5.3.2.1Tablas y cadenas.....	Pág. 75
5.3.2.2Destino de las reglas.....	Pág. 75
5.3.2.3Estado y seguimiento de las conexiones.....	Pág. 76
5.3.2.4Parámetros más usados de iptables.....	Pág. 76
5.3.3Implementación genérica con DROP por defecto.....	Pág. 77
5.4.openBSD - Packet Filter (pf).....	Pág. 78
5.4.1Plataforma.....	Pág. 78
5.4.2Objetivos del Proyecto OpenBSD.....	Pág. 78
5.4.3Packet Filter.....	Pág. 79
5.4.4Sincronización de reglas y redundancia:.....	Pág. 80
5.4.5Proxy TCP Syn.....	Pág. 81
5.4.6Priorizando ACK con ALTQ (alternate Queueing).....	Pág. 82
5.4.7Normalización de paquetes irregulares.....	Pág. 83
5.4.7.1Ejemplos de utilización de scrub:.....	Pág. 84
5.4.8Implementación de un firewall básico con pf.....	Pág. 84
5.4.9Resumen de uso del firewall.....	Pág. 85
5.5.Conclusión.....	Pág. 86
6.Seguridad Wi-Fi.....	Pág. 87
6.1.Wardriving.....	Pág. 87
6.1.1Relevamientos publicados.....	Pág. 87
6.2.Seguridad en las comunicaciones inalámbricas.....	Pág. 88
6.2.1Método de cifrado WEP.....	Pág. 89
6.2.2Deficiencias de la encriptación WEP.....	Pág. 89
6.2.3Prueba de laboratorio.....	Pág. 90
6.2.4Debilidades del algoritmo Key Scheduling de RC4.....	Pág. 91
6.2.5Método de cifrado WPA.....	Pág. 91
6.3.802.11i.....	Pág. 92
6.4.Consideraciones generales.....	Pág. 92
6.4.1Ocular ESSId y Beacon frames.....	Pág. 92
6.4.2ACLs basadas en MACs.....	Pág. 92
6.4.3Acotar el espacio de red.....	Pág. 93

6.5.Conclusión.....	Pág. 93
7.Conclusión General.....	Pág. 94
8.Anexo I – Línea de tiempo de Evolución de Sistemas Criptográficos.....	Pág. 95
9.Anexo II – Técnicas de evasión de Portsentry.....	Pág. 96
10.Anexo III - Formato de encabezado de paquetes.....	Pág. 98
10.1.Encabezado de paquetes IP.....	Pág. 98
10.2.Encabezado ICMP.....	Pág. 101
10.2.1Lista (Tipo/Código) de mensajes de control más comunes:.....	Pág. 101
10.2.2ICMP Echo Request.....	Pág. 102
10.2.3ICMP ECHO Reply.....	Pág. 102
10.2.4Encabezado TCP.....	Pág. 103
10.3.Encabezado UDP.....	Pág. 104
10.4.Encabezado ARP.....	Pág. 104
11.Anexo IV – Técnicas de evasión para la detección remota del SO.....	Pág. 106
12.Anexo V - Herramientas libres para seguridad Informática y de Redes.....	Pág. 109
12.1.Spyware.....	Pág. 109
12.2.Antivirus.....	Pág. 110
12.3.Anti-Rootkit.....	Pág. 111
12.4.Firewall.....	Pág. 112
12.5.Protección de Email.....	Pág. 113
12.6.Utilidades Web.....	Pág. 114
12.7.Redes.....	Pág. 115
12.8.Sistemas de detección de intrusos.....	Pág. 117
12.9.VPN (Virtual Private Network).....	Pág. 117
12.10.Archivos Temporales.....	Pág. 118
12.11.Wireless.....	Pág. 119
12.12.Encripción.....	Pág. 120
12.13.Otros.....	Pág. 121
13.Anexo V – Sitios de Noticias de Seguridad.....	Pág. 123
Bibliografía.....	Pág. 127
Glosario.....	Pág. 130

Prólogo

Desde tiempos recónditos se sabe que la calidad se encuentra directamente ligada a historia de la humanidad, ya sea cuando el hombre comenzó a construir sus primeras armas, a fabricar sus viviendas, o a elaborar cada uno de los productos de uso general siempre observando sus características y procurando mejorarlas para lograr una generación superior.

Casi 1700 años antes de Cristo regía el código de Hammurabi [HAMMURABI] que legislaba la construcción de casas cuya regla número 229 [HAMMU229] establecía que *"si un constructor construye una casa y no lo hace con buena resistencia y la casa se derrumba y mata a los ocupantes, el constructor debe ser ejecutado"*.

También los Fenicios utilizaron un sistema de acción correctiva para asegurar la calidad con el objetivo de eliminar los errores cometidos.

Pasado el tiempo, los objetivos fueron cambiando colocando como premisa la producción sobre todas las cosas. Junto con los aportes de Taylor [WIKI_TAYLOR] la inspección y la producción se separaron y con la evolución de los métodos las organizaciones fueron convergiendo en mejoras para una mayor y mejor producción, donde varios operarios participaban de la elaboración de un producto, introduciendo procedimientos específicos para atender la calidad de los mismos manteniendo la producción masiva.

Con las grandes necesidades de producción en masa durante la segunda guerra, se originó el control estadístico de calidad basados en muestreos y gráficas de control. Es por esto que el término de calidad ha cambiado durante la historia. Hasta los años 1960, el término de calidad había estado enfocado en corregir las actividades, no prevenirlas. Fué allí donde un equipo japonés fijó los principios básicos de control de *calidad total* [CALIJAPO97], el cual se tiene que dar en todas las áreas del sistema, desde la etapa de diseño hasta las ventas y ejecutando el esquema desde el presidente de la compañía hasta el último de los obreros. Lo mismo debería ocurrir con los procesos informáticos, donde no debe considerarse el manejo de la seguridad de la información como caso final y excepcional, sino que se debería hacer una cuestión de costumbre sostener esta actitud pro-activa hacia la misma, actitud que ni siquiera tendría que ser

“recordada” o “pensada”, sino ejecutada como un acto reflejo, así como se consigue asimilar el conducir un automóvil, ya que es con la incorporación de la rutina y el automatismo en cada momento, que se logran aplicar los procedimientos de prudencia y pro-actividad necesarias todos los días.

1. Introducción y conceptos previos

1.1. Historia

La necesidad de seguridad es un requerimiento relativamente nuevo, ya que hasta hace poco más de veinte años la mayoría de las computadoras no estaban conectadas en red o como caso extremo, varias terminales bobas se relacionaban con un único *mainframe* desde donde se administraba y controlaba todo el sistema.

En la década del ochenta, la combinación del desarrollo de la PC (computadora personal), la creación de un protocolo estándar de red, la caída de los costos de hardware y el desarrollo de nuevas y mejores aplicaciones hicieron que la arquitectura de red sea aceptada por la mayoría de los usuarios. Como resultado LANs y WANs, junto con la computación distribuida generaron un notable incremento en la complejidad de los ambientes computacionales [CANA01] [NETSECMai].

En principio, los desarrollos para sistemas LAN⁸ eran relativamente seguros ya que se encontraban físicamente aislados y de esta manera estaban naturalmente protegidos al permanecer independientes de otras redes privadas. Además, estas redes se encontraban administradas de manera centralizada y con acceso limitado. Para conectar dos puntos había que utilizar una conexión dedicada (P2P¹³) y si se decidía agregar a ese circuito un tercer lugar, era necesario establecer vínculos entre cada una de las partes como se muestra en la figura 1.1 .

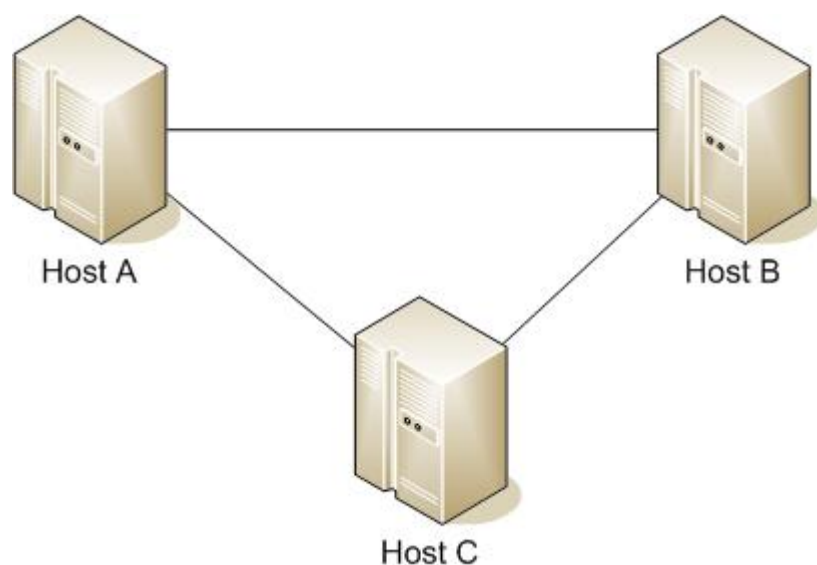


Figura 1.1 - Point-to-point WAN

Con el desarrollo de los protocolos de conmutación de paquetes como el X.25 y el TCP/IP se redujo el costo de las WAN²¹ ya que permitía a muchos sistemas establecer circuitos compartidos. De esta forma múltiples personas y organizaciones podían estar conectadas sobre esta red compartida donde ya no era necesario conectar los sistemas con una configuración de punto a punto[CANA01].

Las vulnerabilidades se introdujeron con el despliegue de este entorno distribuido utilizando redes compartidas, que se vieron agravadas cuando se conectaban LANs aparentemente seguras a través de WANs inseguras. En la figura 1.2 se ilustra el concepto de la red de conmutación de paquetes donde, básicamente una conexión de red de una organización entra en una nube de red *packet-switch* en donde otras organizaciones comparten dicha nube y donde los paquetes de todos ellos se entremezclan [NETSECMai].

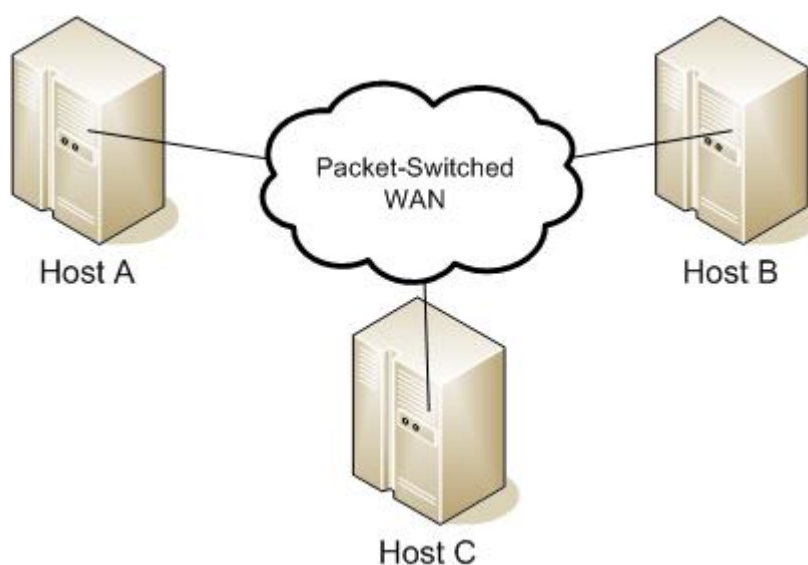


Figura 1.2 - Packet-switched WAN

En este entorno distribuido se puso demasiado énfasis en proveer fácil acceso y buena conectividad, dejando la seguridad como tarea para último momento. Como resultado hubo muchos sistemas abiertos a accesos no autorizados y otros problemas debidos a vulnerabilidades que ahora surgían y nunca antes habían sido notadas.

1.2. Justificación y objetivos

Entonces, teniendo en cuenta lo mencionado, no se puede dejar de considerar la seguridad de la red y el sistema operativo, si se pretende que la información que hasta hoy era considerada privada, no caiga en manos equivocadas, o que se encontrase sorpresivamente el sistema de información inutilizado por la acción de algún código malicioso que pretende tomar el control de la máquina o simplemente sacarla de circulación, por ejemplo, generando procesos basura. Se detallarán las pautas necesarias para poder conseguir un nivel de seguridad aceptable protegiendo el sistema de las debilidades más comunes que día a día se hacen públicas a través del mismo sistema de red que les da vida. Con este trabajo no se pretende construir un instructivo camuflado de cómo hackear una máquina para personas poco capaces, sino hacer comprender que se puede incrementar el nivel de seguridad de una computadora trabajando de forma ordenada y con varios pequeños cambios, para así poder lograr sistemas más fiables y robustos [CANA01] [NETSECMai].

1.3. El primer gran incidente de seguridad informática

Han pasado más de veinte años desde que las primeras computadoras empezaron a interconectarse para crear lo que hoy día forma *la Internet*. Casi el mismo tiempo en que Robert Tappan Morris protagonizaría el primer gran incidente de seguridad informática [WIKI_MW].



El 3 de noviembre de 1988, a los 23 años de edad y recién graduado en la universidad de Cornell, Morris lograría difundir un *worm* o *gusano*²² a través de seis mil servidores Unix (VAX y SUNs) conectados a la red, dejándolos inutilizados por varios días, víctimas de una sobrecarga de tareas invisibles, y originando pérdidas por varios millones de dólares. En ese tiempo ninguno de los usuarios era consciente de que estaba siendo testigo de lo que ha llegado a ser un mito dentro de la comunidad informática [HIS_MOR], esas 99 líneas de código [GUSA_SRC] que causaron el desastre y que luego se convertirían en el famoso *Gusano de Morris*¹.

¹ Finalmente el 22 de enero de 1990 Morris, fue enjuiciado y condenado en la corte de Syracuse, estado de Nueva York, a 4 años de prisión y el pago de US \$ 10,000 de multa, pena que fue conmutada a libertad bajo palabra y condenado a cumplir 400 horas de trabajo comunitario.

Para reproducirse, este gusano aprovechaba una vulnerabilidad en los sistemas operativos UNIX en las plataformas VAX y Sun Microsystems (la herramienta Sendmail, que sirve para el envío y recepción de correos electrónicos, para ser más exactos), con el que era capaz de recolectar contraseñas de los sistemas afectados y así lograr sus propósitos.

El origen de los *gusanos* deriva de los años 60, en los laboratorios de AT&T Bell donde se originó el juego “Core Wars” [COREWA1] [COREWA2] [CARROLLIA]. Este juego consistía en crear un programa que al reproducirse fuera ocupando toda la memoria hasta que en un punto borrara de ella el programa del contrincante². El programador que lograra hacerse con toda la memoria, o que pasado un tiempo obtenga el mayor número de reproducciones, ganaba la partida. La relación entre el gusano de Morris y estos juegos parece clara, sobre todo si se analizan los nombres de los tres estudiantes que dieron origen a “Core Wars”, nada menos que H. Douglas McIlroy, Victor Vysotsky y **Robert Morris** (padre de Robert Morris, creador del gusano y uno de los desarrolladores del UNIX).

1.4. ¿Qué es la seguridad?

La seguridad es una característica que se le pretende dar a un determinado sistema no por considerar que se carece de peligros, sino a partir de afrontarlos y “vencerlos” sin que el sistema “se rompa” (de la misma forma que valentía no es ausencia de miedo, sino enfrentar el miedo con éxito), ya que siempre existen amenazas, conocidas y sobre todo desconocidas. Para poder garantizar seguridad (según comentan la mayoría de los expertos), es necesario concentrarse en hacer que el sistema a proteger será fiable y de manera costo-efectiva y no que sea seguro ciento por ciento, ya que nada lo es. De aquí surge la necesidad de garantizar la integridad, confidencialidad, disponibilidad y autenticidad de los datos que fluyen a través de la red como requisitos fundamentales y pilares de la solución [CANA01] [NETSECMai].

Con **integridad** de datos nos referimos a la completitud y corrección de los mismos haciendo que la única forma de modificarlos, sea de una manera controlada y

2 El juego CoreWar fue desarrollado en Assembler Pnemónico, conocido como Red Code (código rojo) Puestos en funcionamiento, los contendores del CoreWar (ambientes) ejecutaban programas que iban escribiendo la memoria, tanto auto-copiándose, como escribiendo código maligno, disminuyendo la memoria del ambiente a los programas enemigos, hasta que el ganador era el que finalmente conseguía eliminar totalmente el código de los programas enemigos, ocupar toda la memoria disponible, o generar mayor cantidad de copias “auto-replicables”.

por elementos autorizados. Dicha modificación incluye escritura, modificación, cambios de estado, eliminación, creación, bloqueo y reenvío de los mensajes transmitidos.

La **confidencialidad** es la propiedad que asegura que la información existente en un sistema de computación o aquella que es transmitida, sea leída o entendida únicamente por personas o entidades autorizadas. Esto se puede mantener, por ejemplo, con la utilización de llaves simétricas (que será explicado en el Capítulo 3) usando algoritmos de implementación libre como los que se hallan en OpenPGP o GPG⁵ [SIHA95].

Garantizar la **disponibilidad** indica que los datos o objetos estarán accesibles por entidades autorizadas cuando éstas las requieran, confiando en la autenticidad que dichos objetos son los que dicen ser.

Por último, la **autenticación** se logra asegurando que el origen de un mensaje o documento digital es correctamente identificado, lo cual asegura que la identidad provista en el documento respecto de quién fue su creador, no es falsa [NETSECMai].

1.5. Modelos de seguridad

1.5.1 Seguridad por oscuridad

La seguridad por oscuridad [CANA01], a veces llamada seguridad por ocultación, es un método que utiliza el secreto de diseño o implementación para asegurar que básicamente, por desconocimiento, no “se encontrarán” los puntos débiles de dicho sistema. Cabe resaltar el conocido ejemplo de seguridad por oscuridad donde se menciona a los dueños de una casa que guardan una copia de la llave de entrada debajo del felpudo, medida utilizada para contrarrestar el caso de quedar “atrapados” fuera de casa por culpa de un olvido o pérdida de la llave de uso común. La vulnerabilidad teórica sería que alguien entrara a la casa con la copia de la llave escondida. Pero los dueños de la casa creen tener bajo control el tema de que la localización de la llave de repuesto no es conocida públicamente. Mantener este fundamento implica que la revisión de seguridad involucra solo unas pocas personas y a su vez confirma que si se llegara a conocer alguna vulnerabilidad en el sistema, sería

fácilmente atacable por todos los conocedores de esta falla.

1.5.2 Seguridad del perímetro

La seguridad basada en la defensa perimetral apunta a reforzar los puntos de acceso o conexión de nuestra red privada con la red externa. Para ello se tiene que evaluar y planear qué tipos de acceso requiere el sistema, implementar sistemas de seguridad para bloquear el resto del tráfico (por ejemplo *firewalls*⁴ o *proxys*¹⁵), proteger esos únicos puntos vulnerables, y ahí mismo ubicar sistemas de monitoreo y detección de intrusos para que den aviso al administrador del sistema y así poder ejecutar acciones defensivas a tiempo. Cada computadora que se mueve o conecta y desconecta sin pasar por los accesos “oficiales” son el equivalente a *puertas traseras*¹⁶ por las que puede vulnerarse la seguridad de la red [NETSECMai].

Ahora, ¿qué impide que el sistema se “rompa” desde adentro?. Absolutamente nada. Hace años que el perímetro demarcado de nuestra red es demasiado difuso. Las personas conectan computadoras portátiles, PDAs o celulares en otras redes, como el ADSL de su casa, o la conexión wireless de algún café, y luego regresan al interior de la red corporativa listos a vulnerar el perímetro. Esto muestra que no se puede confiar en un único método para proteger el sistema, sino que debe implementarse una defensa de múltiples capas (múltiples perímetros) como si se tratara de una cebolla, para que la solución sea más fiable, además de que en cada componente de la red deben existir medidas de seguridad adicionales [VIHU02].

1.5.3 Seguridad en profundidad

La seguridad en profundidad [COGE04] asume que cada una de las medidas tomadas pueden ser rotas por algún atacante. Sin embargo, a medida que se agreguen capas en el sistema de seguridad, la probabilidad de que el atacante pueda esquivar todas y cada una de ellas sin ser descubierto disminuye proporcionalmente. Esta metodología está basada en un conjunto de reglas cada vez más restrictivas a medida que el objeto a defender se encuentre más cercano. Estas medidas están delimitadas por áreas o zonas de seguridad consideradas de la siguiente manera:

- **Área de influencia:** Es la zona más externa del sistema, donde es factible realizar acciones contra la integridad de ésta área.
- **Área de exclusión:** Es el espacio concéntrico exterior al área protegida, de utilización restringida o acceso limitado.

- **Área protegida:** Es el espacio delimitado por barreras físicas en el que se ejerce un cierto control de movimientos y permanencia.
- **Área crítica:** Es el espacio delimitado por barreras físicas, en el interior del área protegida, cuyo acceso y permanencia son objeto de especiales medidas de control.

1.6. ¿Qué asegurar?

Un sistema informático está compuesto por tres partes: el hardware, el software y los datos. A cualquiera de estos tres componentes pueden ser dirigidos los ataques, y los mismos pueden ser separados según su taxonomía, en: interrupción, interceptación, modificación y fabricación [NETSECREP].

- **Interrupción:** Existe cuando un recurso de un sistema es destruido, o se hace indisponible o inusable. Es un tipo de ataque a la **disponibilidad** del recurso.
- **Intercepción:** Ocurre cuando una parte no autorizada logra capturar el objeto siendo éste, un receptor no autorizado. Esta parte podría ser una persona, un programa o una computadora. Ejemplos incluyen conectar dispositivos a una red de datos privada para capturar paquetes de transmisión, o el copiado ilegal de archivos y programas.
- **Modificación:** Esto se logra cuando además de ganar el acceso al recurso, se modifica y reenvía al destino. Este ataque es a la integridad de los datos, donde por ejemplo se podrían cambiar valores en un archivo o una base de datos, alterar un programa para que funcione distinto, o modificar los mensajes transmitidos en una red.
- **Fabricación:** Es cuando una parte no autorizada genera objetos en el sistema. Se considera un ataque a la autenticidad, y como ejemplos se puede enumerar el agregado de filas o registros a una base de datos, la inserción de paquetes a una red, o en mayor escala, el envío de mensajes desde un servidor de correo electrónico controlado, mediante el reemplazo de la identidad de origen del remitente [NETSECMai] [COGE04].

Todos estos ataques se pueden realizar aprovechando distintas vulnerabilidades en cualquier parte del sistema. Los más comunes los podemos clasificar en:

- **Bugs:** El bug es un error de software generado durante el proceso de creación del mismo cuando no se contemplan todos los posibles estados que el sistema puede tomar en tiempo de ejecución. Los errores más comunes pueden ser: división por cero, un ciclo infinito, desbordamiento de buffer (buffer overflow y underflow), utilización de variables no inicializadas, acceso a un área de memoria restringida, desbordamiento de stack, entre otros. Un ejemplo concreto, y más que conocido es el llamado Y2K o problema del año 2000, causado porque la prioridad de los desarrolladores en las décadas del 60, 70 y 80 (antes de la llegada masiva de las

PC y del microprocesador, cuando la mayoría de las aplicaciones corporativas fueron desarrolladas inicialmente) era que se usara la menor cantidad de memoria que fuera posible, dado que los ordenadores con los que trabajaban las organizaciones eran mucho menos potentes que los primeros computadores personales. La mayoría de los desarrolladores pensaba que dichos sistemas corporativos **"nunca serían usados más de diez años"**. De hecho, algunos desarrolladores habían realizado las rutinas de cálculo de fechas basadas en el almacenamiento de un solo dígito para el año, y por supuesto había graves problemas con el cambio de cada década. Es para tener muy en cuenta que muchos de estos sistemas corporativos siguen funcionando aún hoy, algunos luego de 50 años. Actualmente las prioridades de desarrollo de aplicaciones son otras: flexibilidad, seguridad, robustez, claridad, interconectividad.

- **Backdoors:** También llamado “puerta trasera”, es una secuencia especial en el código, generada por el programador, para saltar el normal flujo del sistema. Estos métodos de salteo de accesos también pueden ser instalados a partir de programas maliciosos como el conocido Back Orifice³ o el Sony/BMGrootkit⁴.
- **Acceso físico:** Otra forma de acceder a los datos es de forma directa a la terminal, con herramientas de crackeo de passwords de login como el OphCrack o el John the Ripper con los que se puede fácilmente obtener el nombre de cuenta de administrador y su contraseñas, y así lograr tener el control completo de la computadora.
- **Bomba Lógica:** Es uno de los mecanismos más viejos de amenaza computacional. Es una rutina que hace que un programa que funciona adecuadamente, en una fecha, momento determinado o condición específica, “explote”, o generalmente deje de funcionar adecuadamente, pudiendo de manera optativa dañar información. Un ejemplo muy conocido fue el famoso virus *Michelangelo*, el cual afectaba el sector de arranque (*boot sector*), con técnica *stealth*²⁰ y de infección lenta, durante casi un año estuvo infectando ordenadores sin realizar ninguna actividad. El día 6 de marzo de 1992 (*aniversario del artista Miguel Angel –Michelangelo - Buonarrotti*) era el llamado “día D”, que puso en marcha la carga destructiva del mismo.
- **Caballo de Troya:** Es un programa útil o aparentemente útil que tiene comandos o procedimientos ocultos y que cuando es invocado, realiza funciones no queridas o dañinas, generalmente para tomar el control de la máquina.
- **Bacterias:** Son programas que no corrompen explícitamente archivos sino que su función es replicarse a sí mismas, realizando como principal daño, que el sistema se sature (disminuya su potencia computacional, espacio libre de memoria RAM, espacio libre en disco, etc.).
- **Virus:** Tal y como ocurre con los homónimos biológicos, son programas ocultos que se replican y liberan en determinadas fechas, o momentos, una carga letal, que generalmente involucra la destrucción de información en el sistema infectado, desde algún archivo de manera aleatoria, hasta todos los discos del sistema infectado [CRO_VIR].

3 **Back Orifice** es un pequeño programa servidor que se instala en una máquina la cual será controlada remotamente por un programa cliente. Los dos componentes se comunican generalmente usando el puerto TCP 31337 (<http://www.bo2k.com>).

4 **Sony/BMGrootkit:** Esquema de protección anti-copia de Sony BMG Music Entertainment que se instalaba en secreto a modo de rootkit en Sistemas Windows, para permanecer oculto, actuando como spyware, enviando a Sony información sobre el usuario.

- **Spyware:** Los programas espías o *spywares* son aplicaciones que recopilan información sobre una persona u organización sin su conocimiento. La función más común que tienen estos programas es la de recopilar datos sobre el usuario y distribuirlo a empresas publicitarias. Dado que el spyware usa normalmente la conexión de una computadora a Internet para transmitir información, consume ancho de banda, y por lo tanto puede verse afectada la velocidad de transferencia de datos de dicha computadora.
- **Rootkits:** Es una herramienta, o grupo de ellas que tiene como finalidad esconderse a sí misma y esconder a otros programas, procesos, archivos, directorios, llaves de registro y puertos, que permiten al intruso mantener el acceso a un sistema para remotamente comandar acciones o extraer información sensible. Existen rootkits para una amplia variedad de sistemas operativos, como Linux, Solaris o Microsoft Windows. En los sistemas Unix/Linux, existen varias aplicaciones para detectar rootkits, entre los más populares se encuentran el *chkrootkit*⁵ y el *rkhunter*⁶.
- **Worms:** Los *gusanos* o *worms* son semejantes a los virus en el sentido de que se esparcen, pero están orientados a otro tipo de contagio, intentando detectar debilidades en determinados programas que atienden servicios, como pueden ser servidores Web, Ftp, mails, entre otros. Una vez que encuentran una vulnerabilidad, envían un mensaje que produce un mal funcionamiento en el servidor, donde queda una copia del worm, el cual comienza a escanear desde ese servidor otras redes. Generalmente terminan por agotar los recursos computacionales de estos servidores, de toda la red, y a veces de toda Internet.
- **Ingeniería Social:** Proviene del estudio sistemático que realizan los atacantes, de las debilidades humanas. Para ello utilizan técnicas y mensajes que hacen uso de diferentes mecanismos, incluso del subconsciente, y normalmente de sentimientos de avaricia, solidaridad, compasión, ira, actualidad, desinformación, etc. Un clásico ejemplo es el envío de un email con la url modificada, de un aviso de cambio de contraseñas del homebanking, obviamente redirigida a un sitio malicioso dedicado a recolectar usuarios y contraseñas de usuarios bancarios.
- **Spam:** Esta técnica se basa en la recolección de grandes bases de datos de correo electrónico por parte de robots que recorren la web en busca de emails. Es fuente común de posteriores infecciones con virus y otras plagas masivas.
- **Scam:** Es una técnica específica de Spam que normalmente sirve para obtener información privada y claves, para posteriormente acceder a sistemas del usuario y realizar hurto de información o bancaria, basado generalmente en los miedos del atacado.

1.7. Mecanismos de seguridad

Las políticas de seguridad deben estar sostenidas sobre tres pilares fundamentales [NETSECREf] [CANA01]:

- 5 **Chkrootkit** o **Check Rootkit** es un programa que permite localizar rootkits conocidos analizando la base de las firmas de los programas de sistema
- 6 **Rkhunter** es una herramienta que detecta rootkits, backdoors y exploits locales mediante la comparación de hashes MD5 de algunos ficheros directorios por defecto de estos rootkits, permisos incorrectos, archivos ocultos, cadenas sospechosas en los módulos de kernel y mas.

- **Prevención:** Utilizando métodos de autenticación, identificación, control de acceso, transmisión segura, plataformas heterogéneas, sistemas honeypot⁶, etc.
- **Detección:** A través de programas de auditoria como Tripwire⁷ o Nagios⁸, encargados de realizar chequeos de integridad, proveer y presentar información al instante sobre el estado actual del sistema.
- **Respuesta:** Implementando métodos de backups y software de análisis forense (para detectar que hizo el intruso y que vulnerabilidad explotó).



Figura 1.3 – Mecanismos de seguridad.

1.7.1 Enumeración de mecanismos de prevención y detección.

En este apartado se pondrá énfasis en los mecanismos de prevención y detección más usados en sistemas Unix y sus derivados, los que se describen a continuación [CANA01] [VIHU02]:

- **Unicidad:** Consiste en incluir en los datos un número de secuencia, fecha/hora, número aleatorio, o alguna combinación de los anteriores para verificar la integridad de los mismos.
- **Control de enrutamiento:** Permite enviar información por zonas clasificadas y a su vez permite solicitar y establecer rutas alternativas en caso de violaciones de seguridad.
- **Tráfico de relleno:** Consiste en enviar tráfico falso junto con los datos válidos para que el atacante no pueda diferenciar los reales de los espurios. Vinculado directamente con la esteganografía, es una forma de ocultar información en objetos que puedan llegar a pasar desapercibidos (como ocultar un árbol en un bosque).

7 **Tripwire** es una herramienta de seguridad e integridad de datos capaz de monitorear y dar aviso ante cambios específicos de archivos.

8 **Nagios** es un sistema de monitoreo de red a nivel de hosts y servicios que contempla la opción de dar aviso al administrador ante un cambio de estado en el sistema.

- **Cifrado/Encriptación:** Fundamental para garantizar la seguridad de la información. Consiste en aplicar un proceso de transformación a un texto claro mediante un cálculo matemático y así obtener un texto cifrado, inentendible para entidades no autorizadas.
- **Gestión de claves:** Abarca la generación, distribución, almacenamiento, tiempo de vida, destrucción y aplicación de las claves de acuerdo a la política de seguridad aplicada.
- **Cortafuegos:** Son conjuntos de aplicaciones o equipos ubicados entre dos redes que establecen la política de acceso entre las partes.
- **Filtrado de paquetes:** En este caso se realiza una lectura pormenorizada de la cabecera de cada paquete y en función a una serie de reglas se permite el paso de los mismos o dicha trama es descartada. Los mismos pueden ser analizados verificando el protocolo utilizado, las direcciones de origen/destino, o los puertos de origen/destino.
- **Proxy de aplicación:** Es un software encargado de eliminar las conexiones a servicios tales como FTP o Telnet, y permiten únicamente la utilización de servicios en donde se encuentra un proxy. Por ejemplo, si la puerta de enlace posee un proxy solo para las conexiones HTTP y FTP, el resto de los servicios no estarán disponibles.
- **Monitoreo de la Actividad:** Es indispensable para garantizar la seguridad del sistema, mantener monitoreada la actividad de las aplicaciones de seguridad, y así poder detectar a tiempo los ataques a los que puede estar siendo sometido.

1.8. Ejemplos básicos de ataques comunes

1.8.1 Ataque externo contra una BD corporativa

Se puede citar un claro ejemplo[CAHO06] de defensa de una base de datos corporativa ubicada en una red local y protegida por varias medidas de seguridad.

Mucha gente puede pensar que un ataque externo necesariamente debe provenir del *router*¹⁷ de la conexión directa de Internet. Sin embargo, los atacantes externos pueden acechar una base de datos corporativa de varias formas:

- Acceso a la base de datos desde Internet.
- Acceso a la base de datos desde un sistema interno atacado y controlado.
- Acceso a la base de datos desde el servidor web atacado y controlado de la DMZ².
- Acceso a la base de datos desde un ataque mediante un *gusano*.

La protección contra estos ataques externos recae sobre las siguientes capas (mencionadas desde el exterior al interior):

- Capa 1: El perímetro del *router* de Internet.
- Capa 2: El perímetro del *firewall* de Internet.

- Capa 3: El firewall de la *DMZ*⁹.
- Capa 4: IDS de Red.
- Capa 5: *NetFlow*¹¹.
- Capa 6: Antivirus.
- Capa 7: *IDS* de Hosts.

Capa 1: El perímetro del router de Internet

La primer capa de protección contra un ataque es el *router externo*. Un router configurado correctamente, puede evitar que el tráfico externo ingrese a una red haciendo *spoofing*¹⁹ de la red interna. Los *routers* pueden también utilizar listas de control de acceso (ACLs) para permitir, denegar, o simplemente identificar el tráfico. Igualmente, un paquete “fabricado a mano” para un ataque podría potencialmente fijar el bit de ACK¹ en 1, y un router podría creer que este paquete es tráfico de la respuesta a una sesión que se originó dentro de la red.

Capa 2: El perímetro del firewall

Mediante un *firewall*, fácilmente se pueden establecer reglas para hacer cumplir ciertas restricciones y mas. En este ejemplo, el firewall previene cualquier conexión de entrada a la red interna. Así mismo, este permite que los sistemas internos establezcan las conexiones de salida (y permite el tráfico de vuelta solamente para las conexiones iniciadas desde la red interna). También se puede tener un control detallado, habilitando procesos que inspeccionen el tráfico de red en la capa de aplicación. También se puede alterar automáticamente el contenido de cada paquete (como campos de IP) y hacer NAT¹⁰ [MRAN93] [SCHI97].

Capa 3: El firewall de la DMZ

A veces, los usuarios externos necesitan tener acceso a ciertos recursos (como un servidor web, o un servidor de correo) que obligadamente los firewall de la primera línea deben dejar pasar. Estos servidores se ubican en un área que no compromete el resto de la red privada.

⁹ Se está suponiendo que la protección es lo más completa posible y por eso existe una verdadera DMZ o Zona Desmilitarizada o De-Militarized Zone, que dispone de dos firewalls, uno externo y uno interno de proveedores y arquitecturas diferentes.

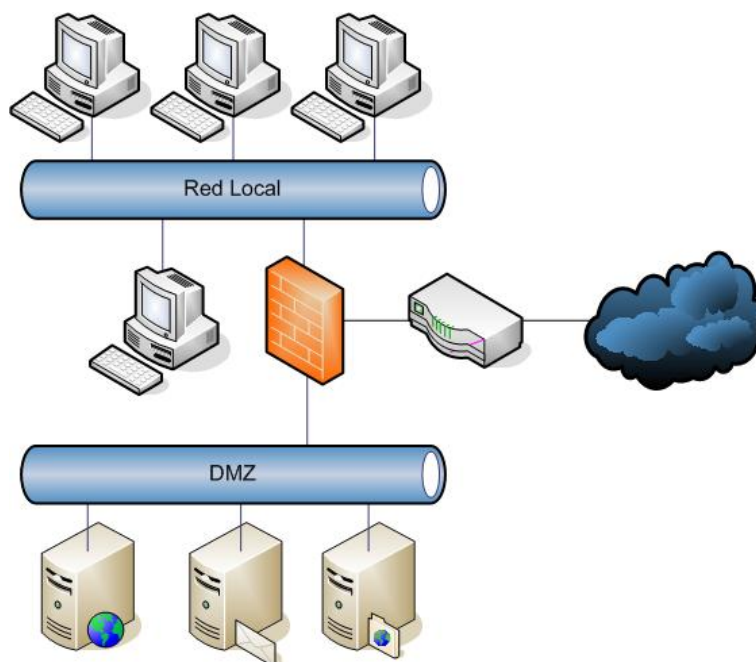


Figura 1.4 – Configuración de DMZ con un único firewall.

A su vez, el firewall esta configurado para evitar el acceso desde la DMZ [WIKI_DMZ] a la red interna para proporcionar una cierta protección contra ataques. A continuación se exponen dos configuraciones típicas de DMZs basadas en el uso de uno o dos firewalls dependiendo el caso.

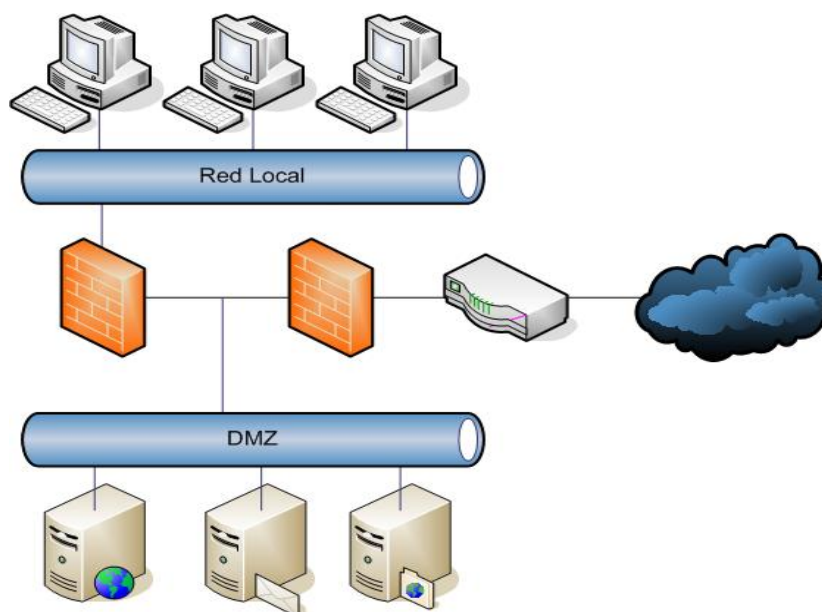


Figura 1.5 – Configuración de DMZ con doble firewall.

Capa 4: Sistemas de Prevención de Intrusos de red

Los sistemas de prevención de intrusos (IPS) son dispositivos ubicados en puntos claves de una red interna, los cuales analizan continuamente el tráfico de la misma en búsqueda de patrones conocidos, guardados en una base de datos, para poder avisar y ejecutar acciones a tiempo para combatir actividades potencialmente maliciosas. En capítulos posteriores se describirán con más detalle diferentes herramientas de este tipo [CAHO06] [COGE04].

Capa 5: NetFlow (Analizadores de tráfico de alto nivel)

Se utiliza para identificar determinados parámetros de una conexión y ubicar patrones conocidos, guardados en una base de datos, de actividades potencialmente anómalas [CISC07]. Los parámetros de conexión que generalmente se apartan son:

- Dirección IP Origen.
- Dirección IP Destino.
- Puerto Origen.
- Puerto Destino
- Protocolo
- Cantidad de Datos Transmitidos.

Capa 6: Antivirus

Otra línea de defensa muy importante se encuentra contenida en el Antivirus, el cual protege al sistema de virus, gusanos y troyanos, especialmente ahora que los sistemas de computadoras superan los límites de nuestras oficinas con el uso de computadores portátiles. Como complemento a este tipo de aplicaciones es recomendable la utilización de programas anti-spyware, anti-rootkit y anti-spam [CRO_VIR].

Capa 7: Sistemas de Prevención de Intrusos en Hosts

Los IPS de host corren de forma local en el sistema a proteger. Poseen mejoras significativas con respecto a los antivirus ya que cuentan con la habilidad de detectar código malicioso como *keyloggers* o *troyanos* desde el mismo instante en que el atacante está instalándolos. Un IPS de host como el Cisco Security Agent (CSA) ,

además de lo anteriormente detallado, protege al sistema de otros tipos de ataque como buffer overflow y posee la posibilidad de alertar al administrador de sistemas mediante el envío de mensajes de texto o e-mails [CAHO06] [COGE04].

1.8.2 Ataque interno contra un Server Administrativo

Otro caso de estudio posible (y altamente probable según varias fuentes [WWWS]) es que el ataque a un servidor sea originado desde la red interna, para lo cual sería de muy poca ayuda proteger el perímetro del sistema si el atacante ya se encuentra en el interior del mismo. Proteger este punto es de vital importancia para garantizar el acceso seguro a los sistemas administrativos. A continuación se describen las diferentes capas a proteger para este ejemplo:

Capa 1: El Switch

La primera capa de protección en una red es el switch en donde se conectan las computadoras. Una buena práctica es separar las diferentes bocas en varias VLANs forzando el tráfico entre las mismas y generando un sistema de protección de capa tres del modelo OSI¹² mediante ACLs en los routers. Con una buena implementación de estas VLANs podemos evitar el uso de ARP spoofing, técnicas usadas por muchas herramientas de captura de tráfico como Ettercap y DSNIFF.

Capa 2: Sistemas de Prevención de Intrusos de red

Como ya se comentó en el ejemplo anterior, estos IPSs se encuentran constantemente analizando el tráfico de la red, buscando patrones de potenciales ataques, los cuales, en algunos casos, se podrían contrarrestar disparando de manera automática acciones o reglas para actuar preventivamente ante este ataque [CAHO06] [COGE04].

Capa 3: Encriptación

Para asegurarse que la comunicación establecida sea privada a los usuarios que la originaron, y contemplando el posible caso de sufrir una captura de tráfico de red, es conveniente utilizar protocolos encriptados como Secure Shell (*SSH*) Secure Socket Layer (*SSL*), los cuales evitan que el tráfico interceptado sea interpretado por el atacante. En este ejemplo, encriptando el tráfico de red resulta casi imposible que un ataque interno obtenga, por ejemplo, las credenciales de logueo y minimiza o

prácticamente elimina la posibilidad de sufrir un ataque del tipo *man-in-the-middle*⁹ en la organización.

Capa 4: Autenticación Fuerte

Este tipo de autenticación se utiliza para proteger activos de alta criticidad para contrarrestar la posibilidad de que alguien utilice la contraseña obtenida de alguna manera (ya sea porque la observó cuando se tipeaba o por medio de alguna aplicación como keyloggers⁷). Una alternativa para implementar esto es utilizar contraseñas del tipo one-time, agregando al típico usuario y clave, una tarjeta Smartcard, un generador de token o una aplicación de software que otorguen un código único para ese logueo.

Un buen ejemplo para citar es el caso de la implementación del e-token³ del home banking del HSBC, un llavero generador de códigos que se modifican constantemente y se utilizan en conjunto con un nombre de usuario (no el número de DNI como en la mayoría de los casos de home banking), una contraseña alfanumérica de más de seis dígitos, y obviamente el código generado por el e-token.

1.9. Recomendaciones Básicas

Para terminar este capítulo, queda claro que no hay una receta única y especial de cómo hacer un sistema seguro, sino que la seguridad es una forma de vida que parte desde el uso consciente de un conjunto de herramientas y costumbres, y esto no se logra de un momento al otro [NETSECMai] [NETSECREf].

Para resumir describiremos una serie de pautas a tener en cuenta por cualquier administrador de redes [KUSE99] [VIHU02]:

- **Contraseñas aceptables:** No está demás que se aclare hasta el cansancio no utilizar contraseñas con palabras de uso frecuente como fechas de cumpleaños, nombres propios o palabras muy cortas. Conviene utilizar contraseñas largas, de por lo menos seis caracteres, con letras mayúsculas y minúsculas, y en lo posible intercalar algún número.
- **Confidencialidad en las contraseñas:** Esta es otra característica que hay que tener en cuenta, no divulgar la contraseña ni tenerla escrita en un lugar a la vista. Reemplazarla cada un tiempo determinado y no utilizar la misma password en los diferentes sistemas que se utiliza (login de inicio, home banking, correo, protector de pantallas).
- **Restricción por usuario de ejecución de programas:** Una de las mejores implementaciones de seguridad es considerar el uso de políticas de grupo en la red local, desde la cual se puede restringir el acceso a instalación y ejecución de aplicaciones por grupos de usuarios.
- **Restricción de booteo de los equipos:** Para que arranquen solo desde el disco rígido adecuado (un sistema que permita el arranque desde otra unidad ofrece una

forma extremadamente sencilla de acceso a los datos del mismo, sobrepasando todas las restricciones de seguridad locales).

- **Utilización de claves de BIOS:** Para mantener a los usuarios alejados de estas.
- Restricción de acceso físico a los servidores. En lo posible utilizar llaves electrónicas con tarjetas y logs de acceso.
- **Mensajes cifrados:** Una costumbre poco utilizada pero de muy fácil implementación es el uso de llaves públicas y privadas para garantizar la comunicación entre dos partes de forma única y segura como GnuPG y PGP.
- **Mantener la fecha/hora sincronizada** con un servidor de tiempo para facilitar la lectura de logs.
- **Sustituir el uso de Telnet y Ftp por Ssh y Scp.**
- **Cerrar los servicios que no sean estrictamente necesarios.**
- **Cambiar el nombre del usuario administrador por default** o mejor así, evitar el login con el usuario root y contemplar la alternativa de usar el comando “sudo”.
- **Utilizar herramientas de monitoreo de equipos de red y sus servicios.** Si es posible coordinado con avisos por email o sms para un aviso instantáneo al administrador del sistema.
- **Mantener las copias de seguridad actualizadas,** duplicadas o triplicadas, y en diferentes edificios.
- **Utilizar versiones del sistema operativo y software actualizadas.**
- **Chequear periódicamente los logs del sistema.**
- **Implementar plataformas heterogéneas** (sistemas operativos, firewalls por software como Iptables y mediante *PIX*¹⁴)
- Y sosteniendo una mentalidad para nada paranoica, ante cualquier actividad sospechosa, **desconectar** el cable de la red externa.

1.10. Conclusión

Ahora si, se está en condiciones de dar una definición de seguridad un poco más completa. Por ello, se define como “Seguridad en redes”, mantener bajo protección los recursos y la información con que se cuenta en la red, a través de procedimientos basados en políticas de seguridad tales que permitan el completo control del sistema, teniendo en cuenta que los usuarios del mismo son parte esencial que no puede ser menospreciada y considerando que la seguridad comienza y termina con personas, por eso es un deber del administrador de redes o encargado de seguridad hacer de estos conceptos una costumbre para los usuarios y así lograr el conocimiento de las ganancias que se obtienen al implementar estas políticas.

2. Seguridad a nivel del Servidor

En este apartado se verán y explicarán una serie de procedimientos para evaluar y monitorear el sistema y su entorno, así como también se recopilará información sobre el mismo para observar el comportamiento de los diferentes servicios y así poder tomar medidas ante una posible falla.

2.1. Verificación de accesos – Syslog¹⁰

En los sistemas Unix y Linux se tiene a disposición una herramienta que guarda los mensajes enviados por los diferentes servicios y a su vez, los generados por el mismo kernel, fundamental para una lectura de análisis. El mismo corre en forma de demonio (*syslogd*) el cual se encarga de capturar los mensajes que envía el sistema y guardarlos en diferentes archivos, según su procedencia, generalmente en el directorio */var/log*. Para lograr mayor seguridad de estos archivos, el syslog puede ser configurado para que los mensajes se envíen a otro servidor y así tener un segundo respaldo de los archivos. Existen herramientas alternativas como el syslog-ng, la cual se describe en un apartado posterior [syslogNg].

2.1.1 Ejemplo de acceso no válido logueado en el auth.log:

```
root@serverSsh:~# tail /var/log/auth.log
Dec 9 11:18:14 serverSsh sshd[455]: Invalid user usuario from clientSsh
Dec 9 11:18:14 serverSsh sshd[455]: Failed none for invalid user usuario from clientSsh
port 4250 ssh2
Dec 9 11:18:17 serverSsh sshd[455]: pam_unix(ssh:auth): check pass; user unknown
Dec 9 11:18:17 serverSsh sshd[455]: pam_unix(ssh:auth): authentication failure; logname=
uid=0 euid=0 tty=ssh ruser= rhost= clientSsh
```

2.1.2 Sugerencias

Como protección de estos archivos necesarios para la obtención de información en un análisis de seguridad, se detallan algunos tips para considerar:

- No utilizar los directorios por defecto para guardar los logs ya que son muy

¹⁰ **syslog** fue desarrollado por Eric Allman como parte del proyecto Sendmail en la década del 80, y al comprobarse que este era muy útil lo empezaron a utilizar otras aplicaciones hasta convertirse en el utilizado por defecto en casi todas las derivaciones de sistemas Unix y Linux y así estar a punto de lograr ser un standard documentado en la RFC 3164 [RFC3164].

conocidos por los atacantes.

- Asegurar que sólo el administrador de la máquina tenga permiso de acceso a estos directorios y archivos .
- Si se envían los mensajes a una máquina remota, tener en cuenta la posibilidad de interceptación de la información .

2.1.3 Alternativas - Syslog-ng [syslogNg]¹¹:

Como alternativa al ya mencionado *syslog*, existe una aplicación de características similares a éste, pero que implementa algunas mejoras. El *syslog-ng* es un sistema de logueo flexible y altamente escalable, ideal para crear soluciones de logueo centralizadas. Algunas de las mejores características se detallan a continuación:

- Transferencia de logs de manera fiable desde el host al servidor o servidores remotos, enviados mediante el protocolo TCP asegurando que no haya pérdidas de mensajes.
- Logueo seguro mediante SSL/TLS¹²: Los mensajes de logs pueden contener información sensible la cual no es conveniente que sea visualizada por terceros. Syslog-ng puede utilizar TLS para encriptar la comunicación. TLS también permite la autenticación mutua entre el cliente y el servidor mediante el uso de certificados X.509.
- Buffering de mensajes en disco¹¹: Si la conexión de red no se encuentra disponible durante el envío de mensajes, el syslog-ng almacena durante este tiempo los mismos en el disco local hasta que la conexión se reponga. De esta manera no existe pérdida de información ante una falla en la red.
- Acceso directo a Base de datos¹¹: Conservación de sus mensajes en una base de datos, lo que permite realizar búsquedas y consultas de los mensajes e interoperar con analizadores de logs. Esta aplicación puede interactuar con MySQL, Oracle, PostgreSQL y SQLite.
- Entornos heterogéneos: Syslog-ng permite recoger los registros de forma masiva en entornos heterogéneos utilizando diferentes sistemas operativos y plataformas de hardware, incluyendo Linux, Unix, BSD, Sun Solaris, HP-UX y AIX.

Además existe un agente el cual puede transferir registros de Microsoft

¹¹ **Syslog-ng** es un demonio de syslog más flexible que el syslog tomado de BSD. Permite aplicar filtros, clasificar de acuerdo a distintos orígenes y enviar a diferentes destinos los logs.

¹² Disponible solo para la versión Premium Edition.

Windows y almacenarlos en un servidor syslog-ng.

- Filtrar y clasificar: Syslog-ng puede ordenar los mensajes de registro basados en su contenido con diversos parámetros, como host de origen, aplicación y prioridad. Filtrados complejos utilizando expresiones regulares y operadores booleanos ofrecen la posibilidad de realizar envíos de los mensajes muy importantes a destinos específicos.
- Soporte de Ipv4 e Ipv6: Además, esta aplicación está preparada para trabajar en ambos entornos de red.

2.1.4 Extensión de características de edición Open Source con Python

Como aporte extra, se plantea el desarrollo de dos aplicaciones en lenguaje de programación Python que se ejecuten de manera independiente para poder agregar características del syslog-ng Premium Edition como son el **Buffering de mensajes** en disco y el **soporte de Base de datos**. De esta manera, un proceso guarda de manera local los mensajes de syslog a modo de buffer (como método de contingencia ante la posibilidad de una eventual caída del servicio de red o la base de datos) y un segundo proceso lee este archivo de buffering y graba los mensajes en una base de datos externa (siempre y cuando exista conexión) [syslogNg].

2.1.4.1 Soporte de Buffering de mensajes en disco

```
#!/usr/bin/python
import sys, signal
def handler(signum, frame):
    salir = True
if __name__ == '__main__':
    outp = open("buffer.log", "a")
    salir = False
    # Se programan las señales a las cuales se va a responder
    signal.signal(signal.SIGHUP, handler)
    signal.signal(signal.SIGTERM, handler)
    signal.signal(signal.SIGKILL, handler)
    try:
        while not salir:
            input_log = sys.stdin.readline()
            outp.write(input_log.rstrip())
    except IOError:
        pass
    outp.close()
```

2.1.4.2 Acceso directo a Base de datos

```
#!/usr/bin/python
import sys, signal
import BDConnection
def handler(signum, frame):
    salir = True
if __name__ == '__main__':
    bd = BDConnection()
    salir = False
    while not salir:
        outp = open("buffer.log", "r")
        input_log = outp.readline()
        outp.close()
        bd.insert(input_log.rstrip())
        outp = open("buffer.log", "w")
        outp.write( '\n'.join(input_log[1:] ) )
        outp.close()
    bd.close()
```

Este código fuente para su uso en un sistema de producción debería contemplar el uso de interrupciones mediante semáforos para que sea confiable. En este caso es presentado con una visión meramente descriptiva.

2.2. Chequeo de tráfico en la red

Estos sistemas de monitoreo se encargan de observar las conexiones dentro de una red y en algunos casos también son los responsables de dar aviso al administrador de la red e incluso bloquear una conexión. Entre las aplicaciones más utilizadas para el monitoreo de tráfico de red se encuentran: [HSRR00]

2.2.1 Netstat

El comando Netstat es una aplicación utilizada para visualizar las conexiones de red al equipo, tablas de enrutamiento y listado de puertos abiertos.

```
root@pablo-laptop:~# netstat
Conexiones activas de Internet (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.218.106.xx:56733     10.218.1.xx:webcache    ESTABLECIDO
tcp        0      0 10.218.106.xx:40905     10.218.1.xx:webcache    ESTABLECIDO
udp        0      0 localhost:32769         localhost:32769         ESTABLECIDO
```

Activar zócalos de dominio UNIX (w/o servers)

Proto	RefCnt	Flags	Type	State	I-Node	Ruta
unix	2	[]	DGRAM		8490	@/com/ubuntu/upstart
unix	13	[]	DGRAM		15188	/dev/log
unix	2	[]	DGRAM		8779	@/org/kernel/udev/udev
unix	3	[]	FLUJO	CONECTADO	107602	/tmp/.esd/socket
unix	3	[]	FLUJO	CONECTADO	107601	
unix	3	[]	FLUJO	CONECTADO	77202	

2.2.1.1 Consulta de conexiones, puertos activos y pasivos

Junto con las opciones `-t`, `-u`, `-w`, y `-x` se muestran las conexiones activas de puertos TCP, UDP, RAW o Unix. Si a su vez se incluye el modificador `-a`, se mostrará también una lista de los puertos que estén esperando una conexión (es decir, que estén escuchando). Esto dará una lista de todos los servicios que estén corriendo actualmente en el sistema.

2.2.1.2 Consulta de Tablas de enrutamiento

Si se ejecuta el comando *Netstat* usando el indicador `-r` se puede observar las tablas de enrutamiento al igual que se verifica con el comando *route*.

```
root@pablo-laptop:~# netstat -r
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window  irtt Iface
10.218.106.0     *                255.255.255.0   U        0 0        0 eth0
169.254.0.0      *                255.255.0.0    U        0 0        0 eth0
default          10.218.106.1    0.0.0.0        UG       0 0        0 eth0
```

2.2.1.3 Consulta de estadísticas de una interfaz

Si se combina con el parámetro `-i`, *Netstat* presenta estadísticas de las interfazs físicas del host.

```
root@pablo-laptop:~# netstat -i
Tabla de la interfaz del núcleo
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0 1500 0 67366 0 0 0 38712 0 0 0 BMRU
eth1 1500 0 0 0 0 0 2015 0 0 0 BMU
lo 16436 0 1464 0 0 0 1464 0 0 0 LRU
```

Con la opción `-a` se muestra el estado de todos los sockets, todas las tablas de enrutamiento, y todas las interfaces lógicas y físicas.

```
root@pablo-laptop:~# netstat -a
```

Conexiones activas de Internet (servidores y establecidos)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:42465	*:*	ESCUCHAR
tcp	0	0	*:nfs	*:*	ESCUCHAR
tcp	0	0	*:52807	*:*	ESCUCHAR
tcp	0	0	localhost:mysql	*:*	ESCUCHAR
tcp	0	0	*:netbios-ssn	*:*	ESCUCHAR
tcp	0	0	*:sunrpc	*:*	ESCUCHAR
tcp	0	0	*:www	*:*	ESCUCHAR
tcp	0	0	localhost:ipp	*:*	ESCUCHAR
tcp	0	0	localhost:postgresql	*:*	ESCUCHAR
tcp	0	0	*:38300	*:*	ESCUCHAR
tcp	0	0	*:microsoft-ds	*:*	ESCUCHAR
tcp	0	0	10.218.106.xx :56733	10.218.1.40:webcache	ESTABLECIDO
tcp	0	0	10.218.106.xx :56731	10.218.1.40:webcache	ESTABLECIDO
tcp	0	0	10.218.106.xx :60239	10.218.1.40:webcache	ESTABLECIDO
tcp	0	0	10.218.106.xx :43672	10.218.1.40:webcache	ESTABLECIDO
tcp	0	0	10.218.106.xx :40904	10.218.1.40:webcache	ESTABLECIDO
tcp6	0	0	*:ssh	*:*	ESCUCHAR
udp	0	0	*:32768	*:*	
udp	0	0	*:nfs	*:*	
udp	0	0	localhost:32769	localhost:32769	ESTABLECIDO
udp	0	0	*:32770	*:*	
udp	0	0	*:32771	*:*	
udp	0	0	10.218.106.xx :netbios-ns	*:*	
udp	0	0	*:netbios-ns	*:*	
udp	0	0	10.218.106.:netbios-dgm	*:*	
udp	0	0	*:netbios-dgm	*:*	
udp	0	0	*:bootpc	*:*	
udp	0	0	*:715	*:*	
udp	0	0	*:sunrpc	*:*	

Activar zócalos de dominio UNIX (servidores y establecidos)

Proto	RefCnt	Flags	Type	State	I-Node	Ruta
unix	2	[ACC]	FLUJO	ESCUCHANDO	16909	/tmp/.X11-unix/X0
unix	2	[ACC]	FLUJO	ESCUCHANDO	18862	/tmp/keyring-2023Sn/socket
unix	2	[ACC]	FLUJO	ESCUCHANDO	19306	/tmp/ssh-GYbxXD6770/agent.6770
unix	2	[ACC]	FLUJO	ESCUCHANDO	20183	/tmp/.ICE-unix/6770
unix	2	[ACC]	FLUJO	ESCUCHANDO	19216	/var/run/netapplet.socket
unix	2	[ACC]	FLUJO	ESCUCHANDO	21009	/var/run/sdp
unix	2	[ACC]	FLUJO	ESCUCHANDO	20408	/tmp/.esd/socket

Los campos corresponden al protocolo usado por el socket (tcp, udp, raw), bytes no copiados por el programa conectado a ese socket, bytes no reconocidos por el host remoto, dirección/puerto del extremo local del socket, dirección/puerto del extremo remoto del socket, estado del socket¹³ y usuario Uid del usuario propietario del socket.

13 - ESTABLISHED: El socket ha establecido conexión

- SYN_SENT: El socket está intentando establecer conexión

2.2.2 Ntop

Ntop (Network Top) [WIKI_nTop] es una herramienta que permite monitorear en tiempo real usuarios y aplicaciones que estén consumiendo recursos de red y a su vez puede dar aviso de errores¹⁴ visualizados con un alerta del lado del cliente de la aplicación. Posee una interfaz la cual se accede vía web a través del puerto 3000 o de forma segura por medio del puerto 3003, desde donde se pueden obtener estadísticas mediante un conjunto de gráficos, los cuales agrupan las conexiones por protocolos, usuarios, etc. Es una aplicación fundamental para el administrador de redes que permite observar como está siendo utilizado el ancho de banda para verificar posibles abusos, vulnerabilidades o fallas en el esquema de red. Estos reportes se encuentran en forma de gráficos y resúmenes de tráfico histórico, detalle agrupado por origen del mismo (local, remoto, o ambos), sentido (Solo enviado, solo recibido, o ambos), filtro por protocolos y una serie de utilidades para exportar las estadísticas de tráfico recopiladas por *ntop* en forma de archivos de texto, XML y otras [KUSE99].

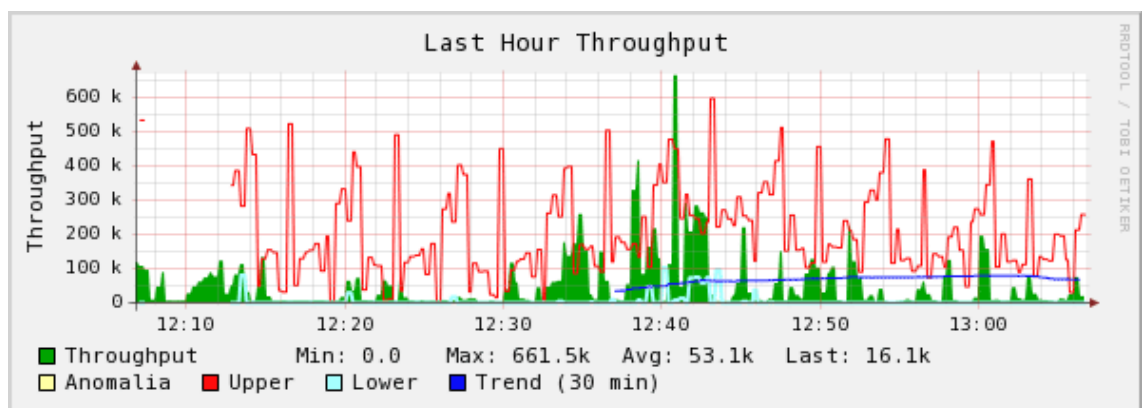


Figura 2.1 - Ntop

-
- SYN_RECV: Una solicitud de conexión ha sido recibida desde la red
 - FIN_WAIT1: El socket ha sido cerrado y la conexión está siendo cerrada
 - FIN_WAIT2: La conexión se cerró y el socket está esperando ser cerrado por la finalización remota
 - CLOSED: El socket no está siendo utilizado
 - TIME_WAIT: El socket está esperando para enviar paquetes a la red
 - LISTEN: El socket está esperando conexiones entrantes

- 14 - Uso de IP's duplicadas
- Identificación de hosts locales en "modo promiscuo"
 - Fallas en la configuración de aplicaciones analizando el protocolo de tráfico de datos
 - Detección de uso inapropiados de los servicios, como hosts que no utilizan el proxy especificado
 - Identificación de hosts que utilicen protocolos que no son necesarios
 - Detección de estaciones de trabajo que trabajen como routers
 - Utilización excesiva del ancho de banda

2.2.3 Nessus

Nessus es una herramienta diseñada para realizar chequeos de vulnerabilidades conocidas de manera automática y corre sobre múltiples sistemas operativos.

2.2.3.1 Historia

El proyecto *Nessus* fué iniciado y liberado bajo licencia GPL en el año 1998 por *Renaud Deraison*. Gran parte de su éxito se debe a que una gran cantidad de desarrolladores colaboran con este trabajo, sobre todo en la construcción de plugins de detección. En octubre de 2005, *Tenable Network Security* cambió *Nessus3* a una licencia propietaria aunque gratuita, donde se distribuyen los binarios pero no así sus fuentes, es por eso que para correr dicha versión es necesario descargarla y compilarla en muchas de las distribuciones Linux como es el caso de nuestro servidor Ubuntu.

2.2.3.2 Estructura

Está formado básicamente por tres partes que se describen a continuación [HACKPRO]:

- **Servidor:** Es servidor de Nessus (*nessusd*) es el demonio que realiza los escaneos propiamente dichos.
- **Cliente:** El cliente, que puede ser instalado en la misma máquina o en forma remota, es el encargado de realizar la interfaz con el administrador, ya sea en forma gráfica o de consola. Este, al momento de finalizado el escaneo, muestra un reporte que puede ser exportado en varios formatos como XML, LaTeX y HTML, o utilizar la alternativa de generar una base de conocimiento para tomar como referencia en los próximos escaneos de vulnerabilidades.
- **Plugins:** Debido a su interfaz modular, basada en plugins, se puede hacer que esta herramienta realice el chequeo específico de ciertas características de conectividad de servicios y aplicaciones escritos en un lenguaje propio llamado NASL¹⁵ los cuales pueden ser activados o desactivados personalizando, de esta manera, el chequeo a realizar. Hay que tener en cuenta que existen varios plugins que pueden “confundir” al sistema de detección de intrusos que está siendo atacado donde sería posible la activación de un método de defensa que podría generar una eventual caída del sistema.

2.2.3.3 Instalación

Disponible en <http://www.nessus.org>, se deben descargar tanto el cliente

¹⁵ NASL (Nessus Attack Scripting), lenguaje de Scripting optimizado para interacciones personalizadas en redes.

como el servidor¹⁶ e instalar mediante el comando *dpkg*¹⁷.

```
dpkg -i --force-architecture /home/pablo/Desktop/Nessus-3.2.0-ubuntu710_i386.deb
dpkg -i --force-architecture /home/pablo/Desktop/NessusClient-3.2.0-debian4_i386.deb
```

Se debe generar un usuario con el comando *nessus-add-first-user*:

```
pablo@pablo-laptop:~$sudo /opt/nessus/sbin/nessus-add-first-user
```

Para registrar la versión se utiliza el comando *nessus-fetch* junto con un número de serie que llega en el mail de registro:

```
pablo@pablo-laptop:~$/opt/nessus/bin/nessus-fetch --register 7CD7-XXXX-1827-YYYY-1EB4
```

Como es de costumbre en los sistemas Linux derivados de Debian, para arrancar y parar un servicio se utilizan los siguientes comandos:

```
pablo@pablo-laptop:~$sudo /etc/init.d/nessusd start
pablo@pablo-laptop:~$sudo /etc/init.d/nessusd stop
```

Para ejecutar el cliente existe un acceso directo en la barra de *Aplicación*, dentro del grupo *Internet*.

2.2.3.4 Funcionamiento

Como fue mencionado anteriormente, *Nessus* basa su poderío en una base de firmas desarrolladas en lenguaje *NASL*, las cuales están escritas en formas de reglas que matchean con aplicaciones, versiones y vulnerabilidades almacenadas en una base de datos para escanear el sistema y dar aviso ante alguna falencia de seguridad.

Nessus permite realizar una exploración proactiva de los equipos de una red determinada para armar y retornar un reporte relacionado con las siguientes deficiencias [HACKPRO]:

- Utilización de servicios no actualizados y que contienen deficiencias de seguridad conocidas como versiones antiguas de *Apache*, *Mysql* o *Squid*.
- Deficiencias relacionadas con malas configuraciones (muchas veces dadas por

16 Hay que verificar la versión, sistema operativo y arquitectura del paquete.

17 En este caso se usó el parámetro *--force-architecture* ya que fué instalado en un Ubuntu 7.10 64bits cuya versión no estaba disponible.

- instalaciones por default) o configuraciones demasiado permisivas.
- Análisis de implementaciones de la pila TCP/IP del equipo remoto.
- Rastros o archivos generados por default olvidados en instalaciones descuidadas.
- Vulnerabilidades conocidas en servicios y aplicaciones según la plataforma y utilizando exploits documentados.

Para realizar estos análisis, *Nessus* divide los pluggins mencionados en peligrosos o no, dependiendo del daño que puedan llegar a generar en el sistema analizado.

Una vez implementado el sistema, la ejecución del mismo es extremadamente sencilla. Primeramente se debe iniciar el demonio (generalmente con el comando `/etc/init.d/nessusd start`) y luego conectarse con un cliente (que puede ser ejecutado en la misma máquina) desde donde se seleccionan los *pluggins* con los que se realizará el escaneo (algunos de ellos pueden llegar a tener parámetros dependiendo el caso) y los equipos o redes a analizar [HACKPRO].

El reporte generado se puede guardar en diferentes formatos. De fácil lectura, está formado principalmente por un listado de vulnerabilidades encontradas categorizadas según su peligro, con un detalle de su incidencia, en la mayoría de los casos la descripción de un solución básica (que puede ser un upgrade de versión, una eliminación de algún archivo olvidado en una instalación, etc.), un *id de vulnerabilidad* y muchas veces un link a una descripción mucho más detallada.

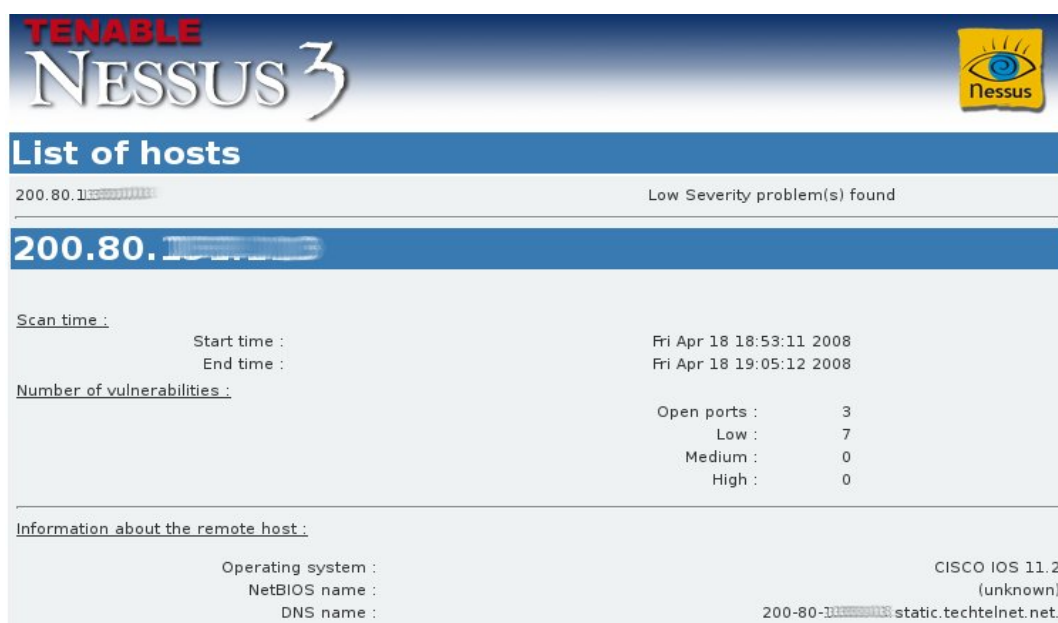


Figura 2.2 – Nessus3

2.2.4 Nagios

El sistema de monitoreo Nagios es una aplicación open source que verifica el estado de los hosts y servicios especificados, y posee la capacidad de alertar (ya sea en forma gráfica, sonora, mediante mails al administrador¹⁸, sms, o incluso un sistema de desarrollo propio) cuando el comportamiento no sea el adecuado y cuando el mismo retorne a su estado normal [PRONAGIOS].

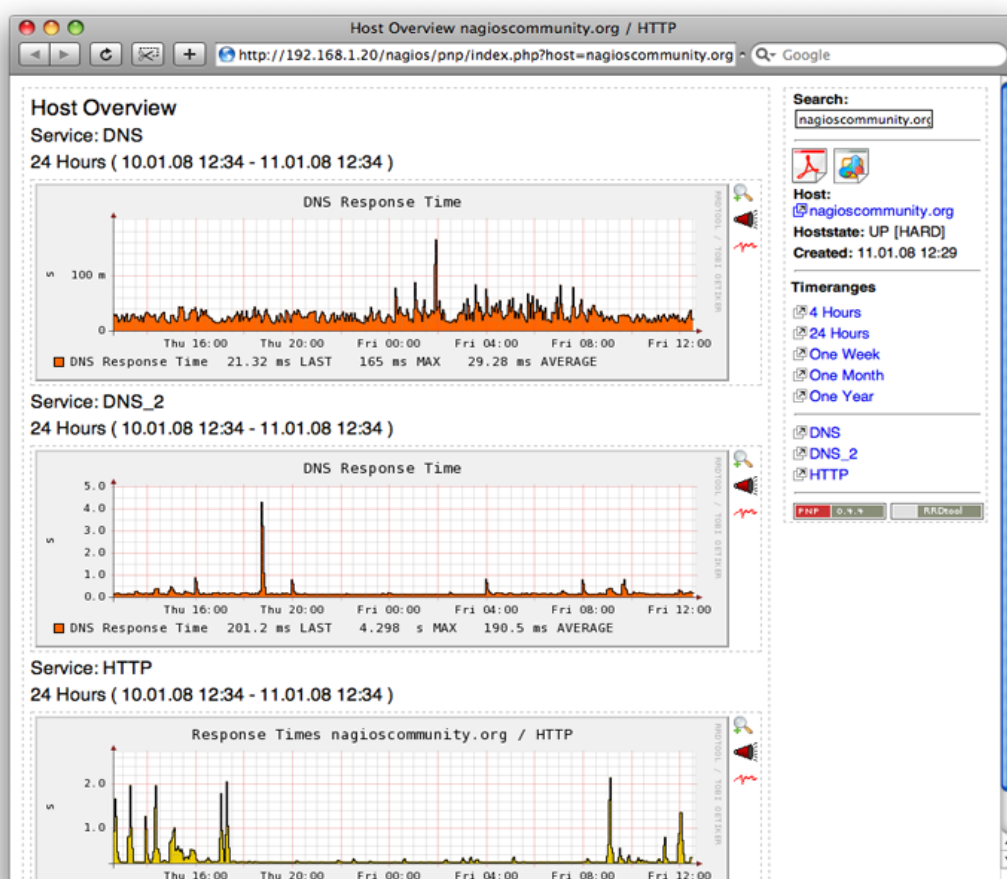


Figura 2.3 – Nagios

2.2.4.1 Instalación

El método de instalación que se utilizó en este caso fue mediante el comando *apt-get* (disponible en las versiones de Linux derivadas de Debian), que nos brinda la versión 2.9-1 la cual hemos testeado, analizado, modificado y usado desde hace más de un año. Desde el sitio oficial, <http://www.nagios.org>, se encuentran

¹⁸ Los mensajes se pueden enviar a personas pertenecientes a diferentes grupos responsables, según la fecha, hora, día de semana, host o servicio alertado.

disponibles los fuentes de la versión estable 3.0, a partir de mediados de marzo de este año, para compilar¹⁹. Al igual que el resto de las aplicaciones, los archivos de configuración de nagios se encuentran disponibles en */etc/nagios2*, donde como primera instancia se puede verificar la configuración del servidor web (Apache 2.x), dentro del archivo */etc/nagios2/apache2.conf*. Para ejecutar el cliente se abre desde un explorador web la dirección <http://localhost/nagios2>. Dicha *url* está protegida²⁰ por el servidor web. Para crear el usuario con permisos se emplea el comando *htpasswd* de la siguiente manera [WWWNAGIOSES] [WWWNAGIOS2]:

```
root@pablo-laptop:/etc/nagios2# htpasswd -c /etc/nagios2/htpasswd.users nagiosadmin
New password: XXXXXXXX
Re-type new password: XXXXXXXX
Adding password for user nagiosadmin
root@pablo-laptop:/
```

Durante la instalación se crearon algunos directorios donde se ubican los principales componentes del *nagios2*:

- ***/etc/nagios2***: Archivos de configuración de la aplicación.
- ***/var/log/nagios2***: Logs de la aplicación.
- ***/usr/sbin/nagios2***: Binarios de la aplicación.
- ***/etc/nagios-plugins/config***: Archivos que definen los plugins de nagios2.
- ***/usr/lib/nagios/plugins***: Comandos que son utilizados por los plugins de nagios2.

2.2.4.2 Configuración

Los archivos de configuración del monitoreo en sí se encuentran en el directorio */etc/nagios2/conf.d/* y es allí donde se establecen los parámetros de qué hosts y servicios se verificarán, en qué período de tiempo, a quién dar aviso y de qué manera. Los principales objetos que maneja nagios2 se enumeran a continuación [PRONAGIOS] [WWWNAGIOSES] [WWWNAGIOS2]:

¹⁹ Para instalar Nagios 3.x a partir de los fuentes, se encuentra disponible la documentación oficial en http://nagios.sourceforge.net/docs/3_0/quickstart.html

²⁰ *AuthType Basic, AuthUserFile /etc/nagios2/htpasswd.users y require valid-user* en el archivo */etc/nagios2/apache2.conf*

- **host:** Unidades físicas como servidores, routers y firewalls.

```
# Generic host definition template
define host{
    name generic-host ; The name of this host
    template - referenced in other host definitions, used for template recursion/resolution
    notifications_enabled 1 ; Host notifications are enabled
    event_handler_enabled 1 ; Host event handler is enabled
    flap_detection_enabled 1 ; Flap detection is enabled
    process_perf_data 1 ; Process performance data
    retain_status_information 1 ; Retain status information across program restarts
    retain_nonstatus_information 1 ; Retain non-status information across program restarts
    register 0 ; DONT REGISTER THIS DEFINITION
    - ITS NOT A REAL HOST, JUST A TEMPLATE!
}
define host{
    use generic-host
    host_name windows2003
    alias NT 2003 Server
    address 192.168.0.10
    check_command check-host-alive
    contact_groups nt-admins
    max_check_attempts 10
    notification_interval 480
    notification_period 24x7
    notification_options d,u,r
}
```

- **hostgroup:** Colecciones de hosts que generalmente tienen alguna característica en común como por ejemplo la ubicación.

```
define hostgroup{
    hostgroup_name winservers
    alias Windows Servers
    members windows2003,windows2000
}
```

- **service:** Servicios que corren en los hosts que pueden incluir tanto servicios en sí como SMTP o HTTP, o métricas como espacio en disco o uso de memoria.

```
define service{
    host_name paginac,paginag
    service_description PING
    is_volatile 0
    check_period 24x7
    max_check_attempts 4
    normal_check_interval 5
}
```

```
retry_check_interval 1
contact_groups web-admins
notification_interval 240
notification_period 24x7
notification_options c,r
check_command check_ping!1050.0,30%!1550.0,60%
}
```

- **servicegroup:** Al igual que los grupos de host, son colecciones de servicios que poseen alguna característica en común.

```
define servicegroup{
servicegroup_name pingservices
alias Ping Services
members paginac,Ping Service,paginag,Ping Service,switch,Ping Service
}
```

- **contact:** Personas u objetos que pueden ser contactados durante un determinado evento.

```
define contact{
contact_name pablo
alias Nagios Super Admin
contactgroups nt-admins ;podemos agregar desde aquí los grupos
service_notification_period 24x7
host_notification_period 24x7
service_notification_options w,u,c,r
host_notification_options d,u,r
service_notification_commands notify-by-email
host_notification_commands host-notify-by-email
email pablo@saru.com.ar
}
```

- **contactgroup:** Colección de contactos que poseen determinada característica en común, como por ejemplo la consultora responsable del host o servicio que se verifica.

```
# 'network-admins' contact group definition
define contactgroup{
contactgroup_name network-admins
alias Network Administrators
members pablo
}
```

- **timeperiod:** Definición de una ventana de tiempo, por ejemplo el tiempo de trabajo en horas (7x24, 8horas, lun-vie, etc.).

```
define timeperiod{
timeperiod_name workhours
alias Standard Work Hours
}
```

```

monday      09:00-17:00
tuesday     09:00-17:00
wednesday   09:00-17:00
thursday    09:00-17:00
friday      09:00-17:00
}

```

- **command:** Comando que puede ser usado durante el proceso de chequeo, como por ejemplo el comando *ping* utilizado para verificar el estatus de un host.
- **hostextinfo:** Presenta información extendida que se mostrará en el cliente como descripción detallada de un host.

```

define hostextinfo{
    hostgroup_name    debian-servers
    notes              Debian GNU/Linux servers
    notes_url          http://webserver.localhost.localdomain/hostinfo.pl?
    host=network1

    icon_image         base/debian.png
    icon_image_alt     Debian GNU/Linux
    vrml_image         debian.png
    statusmap_image    base/debian.gd2
}

```

Host	Service	Type	Time	Contact	Notification Command	Information
flier	Yum Updates	OK	01-11-2008 11:45:06	nagiosadmin	notify-service-by-email	YUM OK: O/S is up to date.
flier	Memory Usage	OK	01-11-2008 11:42:56	nagiosadmin	notify-service-by-email	OK - 308 / 756 MB (40%) Free Memory, Used: 448 MB, Shared: 0 MB, Buffers: 59 MB, Cached: 274 MB
nero	Physical Memory Usage	WARNING	01-11-2008 11:39:06	nagiosadmin	notify-service-by-email	WARNING: physical memory: Total: 0.999G - Used: 663M (64%) - Free: 360M (36%) > warning
ayamon.com	FTP	OK	01-11-2008 11:34:31	nagiosadmin	notify-service-by-email	FTP OK - 10.753 second response time on port 21 [220 ProFTPD 1.3.0 Server (4Admin(tm) FTP Server) [208.64.136.202]]
flier	Memory Usage	WARNING	01-11-2008 11:32:51	nagiosadmin	notify-service-by-email	WARNING - 8 / 756 MB (1%) Free Memory, Used: 748 MB, Shared: 0 MB, Buffers: 46 MB, Cached: 434 MB
ayamon.com	FTP	CRITICAL	01-11-2008 11:24:14	nagiosadmin	notify-service-by-email	No data received from host
flier	Memory Usage	OK	01-11-2008 11:23:54	nagiosadmin	notify-service-by-email	OK - 276 / 756 MB (36%) Free Memory, Used: 480 MB, Shared: 0 MB, Buffers: 62 MB, Cached: 176 MB
temptraxe1	Probe 2	WARNING	01-11-2008 11:18:13	nagiosadmin	notify-service-by-email	Temp Warning: Attic = 39.8 F
temptraxe1	Probe 1	WARNING	01-11-2008 11:13:53	nagiosadmin	notify-service-by-email	Temp Warning: Furnace Room = 80.7 F
task	Backups: MySQL	CRITICAL	01-11-2008 11:12:03	nagiosadmin	notify-service-by-email	CRITICAL: mysql_2008-01-02_07h00m.Wednesday.sql.gz is too old (9d 4h 11m 58s old)
dev1	Backups: MySQL	CRITICAL	01-11-2008 11:11:43	nagiosadmin	notify-service-by-email	CRITICAL: mysql_2008-01-02_07h00m.Wednesday.sql.gz is too old (9d 4h 11m 37s old)
dev1	Memory Usage	WARNING	01-11-2008 11:10:43	nagiosadmin	notify-service-by-email	WARNING - 93 / 1010 MB (8%) Free Memory, Used: 917 MB, Shared: 0 MB, Buffers: 136 MB, Cached: 846 MB
flier	Yum Updates	CRITICAL	01-11-2008 11:09:33	nagiosadmin	notify-service-by-email	CHECK_NRPE: Socket timeout after 10 seconds.
nero	Physical Memory Usage	WARNING	01-11-2008 10:34:06	nagiosadmin	notify-service-by-email	WARNING: physical memory: Total: 0.999G - Used: 675M (66%) - Free: 348M (34%) > warning
flier	Memory Usage	WARNING	01-11-2008 10:28:57	nagiosadmin	notify-service-by-email	WARNING - 73 / 756 MB (9%) Free Memory, Used: 683 MB, Shared: 0 MB, Buffers: 101 MB, Cached: 471 MB
temptraxe1	Probe 4	OK	01-11-2008 10:16:07	nagiosadmin	notify-service-by-email	Temp Ok: Probe 4 = 23.6 F
temptraxe1	Probe 2	WARNING	01-11-2008 10:16:07	nagiosadmin	notify-service-by-email	Temp Warning: Attic = 38.7 F
temptraxe1	Probe 3	OK	01-11-2008 10:13:37	nagiosadmin	notify-service-by-email	Temp Ok: Port 3 = 72.8 F
temptraxe1	Probe 1	WARNING	01-11-2008 10:13:37	nagiosadmin	notify-service-by-email	Temp Warning: Furnace Room = 81.4 F
temptraxe1	Probe 2	CRITICAL	01-11-2008 10:13:07	nagiosadmin	notify-service-by-email	(Return code of 127 is out of bounds - plugin may be missing)

Figura 2.4 – Nagios – Estado de servicios

- **serviceextinfo:** Presenta información extendida que se mostrará en el cliente como descripción detallada de un servicio.

Durante la instalación por defecto se crean archivos de configuración genéricos que luego se pueden ir incrementando y detallando a medida que se recolecta información de la red, pero lo que realmente hace poderoso este sistema es la colección de pluggins que lo componen.

2.2.4.3 Plugins

En este apartado se definen y ejemplifican los plugins más utilizados, pero para ello es necesario marcar la diferencia entre plugins de nagios2 y comandos. Los plugins de nagios se definen en */etc/nagios-plugins/config* y son archivos similares a los que se describieron en el apartado 2.2.4.2 de Configuración, donde mantienen una estructura determinada con un nombre que los define y una línea de comando que se ejecuta. El listado original es el siguiente, agrupados por la definición de comandos [PRONAGIOS] [WWWNAGIOSES] [WWWNAGIOS2]:

```
root@pablo-laptop:/etc/nagios-plugins/config# ls
apt.cfg      disk.cfg      dummy.cfg     ftp.cfg       http.cfg      load.cfg      mysql.cfg
nt.cfg       ping.cfg      real.cfg      ssh.cfg       users.cfg
breeze.cfg   disk-smb.cfg  flexlm.cfg    games.cfg     ifstatus.cfg  mail.cfg      netware.cfg
ntp.cfg       procs.cfg     rpc-nfs.cfg   tcp_udp.cfg   vsz.cfg
dhcp.cfg     dns.cfg       fping.cfg     hppjd.cfg     ldap.cfg      mrtg.cfg      news.cfg
pgsql.cfg    radius.cfg     snmp.cfg      telnet.cfg
root@pablo-laptop:/etc/nagios-plugins/config#
```

Y como ejemplo se muestra un resumen del archivo */etc/nagios-plugins/config/ping.cfg* que contiene la descripción de los plugins que verifican disponibilidad mediante este comando.

```
root@pablo-laptop:# cat /etc/nagios-plugins/config/ping.cfg
# 'check_ping' command definition
define command{
    command_name      check_ping
    command_line       /usr/lib/nagios/plugins/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c
$ARG2$
}

# 'check-host-alive' command definition
define command{
    command_name      check-host-alive
    command_line       /usr/lib/nagios/plugins/check_ping -H $HOSTADDRESS$ -w 5000,100%
```

```
-c 5000,100% -p 1
    }

# 'check-printer-alive' command definition
define command{
    command_name    check-printer-alive
    command_line    /usr/lib/nagios/plugins/check_ping -H $HOSTADDRESS$ -w 5000,100%
-c 5000,100% -p 1
    }

# 'check-switch-alive' command definition
define command{
    command_name    check-switch-alive
    command_line    /usr/lib/nagios/plugins/check_ping $HOSTADDRESS$ -w 5000,100% -c
5000,100% -p 1
    }

# 'check-router-alive' command definition
define command{
    command_name    check-router-alive
    command_line    /usr/lib/nagios/plugins/check_ping -H $HOSTADDRESS$ -w 5000,100%
-c 5000,100% -p 1
    }

root@pablo-laptop:~#
```

Como se mencionó anteriormente, estos plugins utilizan los comandos que se encuentran en el directorio */usr/lib/nagios/plugins* y se enumeran a continuación:

```
root@pablo-laptop:~# ls /usr/lib/nagios/plugins
check_apt      check_disk      check_ftp      check_imap      check_mailq
check_nntp     check_ping      check_simap    check_tcp      urlize
check_bgpstate check_disk_smb  check_game     check_ircd      check_mrtg
check_nt       check_pop       check_smtp     check_time     utils.pm
check_breeze   check_dns       check_hpjd     check_jabber    check_mrtgtraf
check_ntp      check_procs     check_snmp     check_udp      utils.sh
check_by_ssh   check_dummy     check_http     check_ldap     check_mysql
check_nwstat   check_radius    check_spop     check_ups
check_clamd    check_file_age  check_icmp     check_ldaps     check_mysql_query
check_oracle   check_real      check_ssh      check_users
check_dhcp     check_flexlm    check_ifoperstatus check_load     check_nagios
check_overcr   check_rpc       check_ssmt     check_wave
check_dig      check_fping     check_ifstatus  check_log      check_nntp
check_pgsql    check_sensors   check_swap     negate
root@pablo-laptop:~#
```

Los mismos pueden ser ejecutados directamente desde la línea de comandos para su

prueba inicial y posterior implementacion en un nuevo plugin.

```
root@pablo-laptop:~# /usr/lib/nagios/plugin/check_ping
check_ping: Could not parse arguments
Usage:check_ping -H <host_address> -w <wrta>,<wpl>% -c <crtat>,<cpl>%
[-p packets] [-t timeout] [-4|-6]
root@pablo-laptop:~#
```

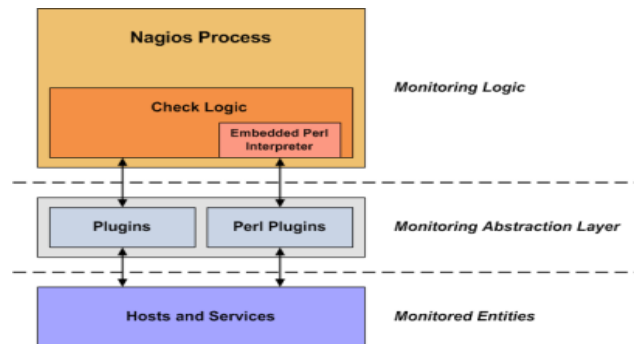


Figura 2.5 – Nagios – Arquitectura de plugins

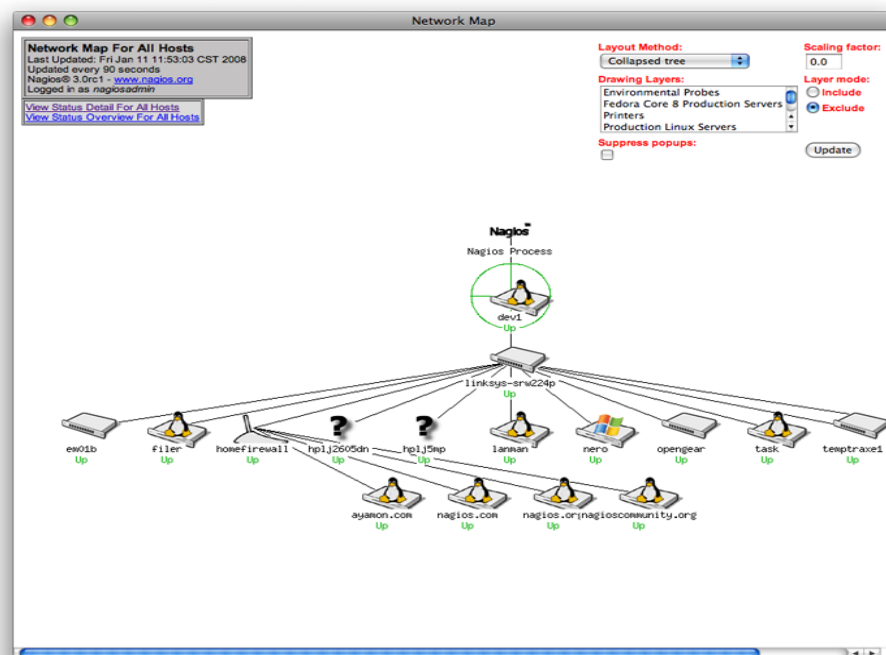


Figura 2.6 – Nagios – Mapa de red

2.2.4.4 Ejecución:

Para un ambiente de producción, es recomendable tener instalado, configurado y en ejecución constante, el *nagios* desde un host dedicado que se encuentre ubicado cercano a la posición del administrador de red, así ante algún evento,

este lo identifica al instante mediante la comunicación en pantalla y los avisos sonoros.

Es una buena práctica armar un mapa de red con la arquitectura propia donde se ubican los hosts en el mismo orden que se encuentran físicamente para una mejor interpretación del estado del mismo [PRONAGIOS] [WWWNAGIOSES] [WWWNAGIOS2].

2.3. Conclusión

Una de las principales necesidades para reaccionar en forma rápida y así contrarestar a tiempo posibles ataques es estar informado sobre la situación actual del sistema en su conjunto, es por eso que durante este capítulo se intentaron plantear diferentes herramientas que se encuentran disponibles para verificación de estados y monitoreo de hosts y servicios, de diferentes vistas y altamente configurables. En el capítulo 4 se analizan las capacidades de otros sistemas de monitoreo como son IDSs e IPSs.

3. Criptografía

Para dar una posible solución a dos de los problemas de la seguridad en el manejo de los datos (privacidad y autenticación), se profundizará sobre los métodos de transformación de los mismos de manera controlada, mediante el empleo de funciones matemáticas, de manera que se pueda garantizar en primer lugar el secreto en la comunicación entre dos puntos, y, en segundo lugar, asegurar que la información que se envía es auténtica en un doble sentido (de manera que el remitente sea quién dice ser, y que su contenido haya llegado sin alteraciones durante su tránsito). Al estudio de estos sistemas que ofrecen medios seguros de comunicación se los conoce como criptología. Estos sistemas de comunicación están basados en mecanismos donde el lado emisor oculta o cifra el mensaje para que solo el receptor autorizado pueda descifrarlo. En la actualidad la criptología ha extendido sus áreas a métodos de autenticación de información digital, comúnmente llamada firma digital [CRIPTO] .

El procedimiento o método utilizado para cifrar datos se realiza mediante la implementación de un algoritmo conocido, el cual puede ser considerado como una función matemática donde entra información en texto claro y es desordenada de manera que la misma se transforme en algo incomprensible a partir de la utilización de al menos un código o clave. En muchos casos esta función de transformación posee su contraparte, la que permite descifrar el mensaje (o sea, mediante el ingreso de un texto cifrado se logra como salida el texto claro), este sistema criptográfico se lo conoce como “de doble vía” o reversible; aunque no siempre es así, y en ocasiones la función de descifrar está ausente, como es el caso del método SHA-1, muy utilizado para el almacenamiento de contraseñas [CRYPTO_ELEC].

Estos métodos de cifrado se encuentran divididos en dos categorías: los llamados “de sustitución” y los “de transposición”. Los primeros, como su nombre lo indica, se logran mediante el intercambio de una letra o grupo de letras, por otra letra o grupo de letras, para disfrazarlas. El cifrador por sustitución más antiguo es el conocido como “Cifrador de César”, atribuido a Julio César, donde la letra A se representaba con una D, la B por una E, la C por una F, y así con cada una de las letras del alfabeto donde son sustituidas por la que se encuentre tres lugares delante de ella (de manera circular, donde luego de la Z vuelve a comenzar con la A). Una variante a este método es permitir desplazar n lugares cada letra donde n es convertida en la clave de cifrado.

También este último método evolucionó cuando se implementó la “Sustitución Monoalfabética”, donde la clave consiste en una cadena completa de caracteres a utilizar relacionados con un carácter del alfabeto. Como se verá en un ejemplo a continuación, se puede llegar a la cadena clave mediante el conocimiento de las propiedades estadísticas de los lenguajes, calculando el porcentaje de ocurrencias de letras en un lenguaje y así deducir la cadena. Los cifradores de transposición en cambio, reordenan las letras en lugar de disfrazarlas. Esto se logra mediante el cambio de posición de los caracteres del mensaje, por ejemplo reescribiendo un texto corrido una cantidad específica de lugares, o mediante en uso de una clave donde se define el tamaño para un vector de cambios.

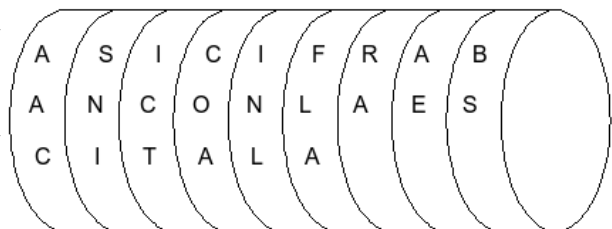
3.1. Historia

Desde las primeras civilizaciones militarizadas se desarrollaron técnicas para enviar mensajes de manera que si el mensajero era interceptado, la información no corra riesgo de caer en manos del enemigo [WIKI_CRYPT]. Uno de los primeros criptosistemas documentados fué descrito por el historiador griego Polibio, quien describió un sistema de sustitución basado en la posición de las letras en una tabla. El sistema de cifra consistía en hacer corresponder a cada letra del alfabeto con un par de letras que indicaban la fila y la columna en la cual aquella se encontraba, en un recuadro de $5 \times 5 = 25$ caracteres, transmitiéndose por lo tanto en este caso el mensaje como un criptograma [CRIPTO] [CRYPTO_KHAN].

Casi al mismo tiempo (unos cincuenta años antes), sus pares romanos utilizaban sistemas de sustitución (el método de César) que actualmente es uno de los más conocidos en la jerga.

Otro de los métodos criptográficos utilizados por los griegos fué la *escítala espartana* [CRYPTO_ELEC], que consistía en un cilindro que servía como clave en el que se enrollaba un mensaje para que

el mismo pueda ser cifrado y descifrado. Una vez escrito el mensaje, la cinta se desenrollaba y era transportada por un mensajero, que en el posible caso de que sea



interceptado por algún enemigo, lo único que este lograría sería un conjunto de caracteres o letras al parecer en forma aleatoria o desordenadas donde la única manera

de leerlo correctamente era enrollándolo en un cilindro de igual tamaño al con que se escribió (en este caso, y como se señaló anteriormente, el diámetro del cilindro era la clave del mensaje).

Ya en 1465, el italiano Leon Batista Alberti inventó otro sistema de sustitución llamado “Polialfabética”, que utilizaba múltiples sistemas de César aplicados periódicamente. Un ejemplo es el sistema criptográfico de Vigenère [WIKI_VIGE] que consiste en una matriz cuadrada la cual contiene 26 alfabetos de César y donde la clave está constituida por una palabra simple más la matriz de 26 x 26. Este sistema resultó seguro debido a la imposibilidad de determinar el largo de la clave. Una vez encontrado el largo de la clave era posible encontrar las sustituciones simples agrupando las letras.

En 1863 F.W. Kasiski [WIKI_KASI] resolvió el problema de encontrar el tamaño de la clave a través de la técnica de “La incidencia de las coincidencias” basada en la búsqueda de palabras repetidas en el texto cifrado.

En los tiempos de Felipe II se utilizaron cifrados basados en cadenas con más de 500 símbolos, método que los matemáticos del rey consideraron en esa época “inexpugnable”, hasta que Francois Viète consiguió criptoanalizarla para el rey de Francia.

Entre el siglo XIX y hasta la segunda guerra mundial, las figuras que se destacaron fueron el holandés Auguste Kerckhoffs²¹ y el prusiano Friedrich Kasiski²², pero fué en el siglo XX cuando la historia de la criptografía vuelve a presentar importantes avances, especialmente durante las dos guerras mundiales, aprovechando herramientas que permitirían conseguir mejores y más seguros cifrados. Una de estas herramientas era la máquina de cálculo alemana *Enigma* [WIKI_ENIG], la cual disponía de un mecanismo de cifrado rotatorio, que permitía utilizarla tanto para cifrar como para decifrar mensajes.

El 17 de enero de 1917 William Montgomery, criptoanalista de la Marina Británica en Londres, interceptó un telegrama lleno de códigos que el Ministro de Relaciones Exteriores alemán Arthur Zimmermann envió a su embajador en los Estados Unidos. El mensaje ocultaba, entre otras cosas, el anuncio de la guerra con los Estados

21 Autor de uno de los libros históricos de la criptografía, *Cryptographie Militaire*, que establece que la seguridad de cifrado debe residir, exclusivamente, en el secreto de la clave y no en el desconocimiento del algoritmo de cifrado.

22 Autor del libro “La escritura secreta y el arte del descifrado”, primer publicación sobre criptoanálisis aplicado a los cifrados de sustitución polialfabéticos basado en el análisis de fragmentos repetidos de texto dentro del texto cifrado, técnica conocida como el método Kasiski.

Unidos. Es así como Estados Unidos entra en la confrontación mundial y ayuda a los aliados a ganar la guerra [CRYPTO_KHAN].

El 7 de diciembre de 1941 interceptan en la radio de la estación naval de Bainbridge Island, un mensaje de nueve minutos desde Tokyo a la Embajada Japonesa en los Estados Unidos. El mismo estaba cifrado con la máquina Purple, y el código roto por William Friedman [WIKI_FRIED]. El mismo anunciaba el ataque de Pearl Harbor.

Tras la finalización de la segunda guerra mundial, durante el año 1948, la criptografía tuvo un gran desarrollo teórico de la mano de Claude Shannon [WIKI_SHANNON], quién logró hacer de ella una rama más de las matemáticas.

A mediados de los años 70, el departamento de Normas y Estándares norteamericano publica el primer diseño lógico de un cifrador, el cual se convertiría a finales del siglo en el Estándar de Cifrado de Datos o DES.

En el año 1976 se publica un artículo desarrollado por Whitfield Diffie y Martin Hellman [WIKI_HELLMAN] de cifrado asimétrico, en el que proponen una nueva filosofía de cifra, dando lugar a los criptosistemas de clave pública.

3.2. Ejemplo del método de Riel, o de Rail Fence

“La seguridad de un sistema no debe depender de mantener en secreto el algoritmo, sino sólo de mantener secreta la clave”

Este método es un sencillo algoritmo de transposición que lo que hace es trazar varias líneas paralelas (siendo la cantidad de líneas la clave) para luego escribir la frase en zigzag sobre ellas [CRYPTOLABS].

```
l---g---d---e---i---m---d---d---n---d---n---e---s---e---l---o---m---n---l---m---e---s---e---a---v---
-a-e-u-i-a-d-u-s-s-e-a-o-e-e-e-d-r-e-a-t-n-r-n-e-r-t-e-a-g-r-t-o-i-o-o-o-e-a-t-n-r-e-r-t-l-c-a-e
--s---r---d---n---t---n---b---p---e---m---e---e---c---o---l---i---s---s---d---n---e---c---a---l---
```

Luego se escriben las letras siguiendo su orden normal en cada línea y el resultado se agrupa y presenta por convención en mayúsculas y agrupadas de a cinco

```
LGDEI MDDND NESEL OMNLM ESEAV AEUIA DUSSE AOEEE EDREA TNRNE
RTEAG RTOIO OOEAT NRERT LCAES RDNTN BPEME ECOLI SSDNE CAL
```

Funciones para encriptar y desencriptar los mismos desarrollado en *javascript*:

```
function Encrypt(plaintext) {
    var key = parseInt(document.getElementById("key").value);
    if(key > Math.floor(2*(plaintext.length-1))){
        alert("Tamaño incorrecto de clave.");
        return;
    }
    ciphertext = "";
    for(line=0; line<key-1; line++){
        skip=2*(key-line-1);    j=0;
        for(i=line; i<plaintext.length;){
            ciphertext += plaintext.charAt(i);
            if((line==0) || (j%2 == 0)) i+=skip;
            else i+=2*(key-1) - skip;
            j++;
        }
    }
    for(i=line; i<plaintext.length; i+=2*(key-1)) ciphertext += plaintext.charAt(i);
    return ciphertext;
}

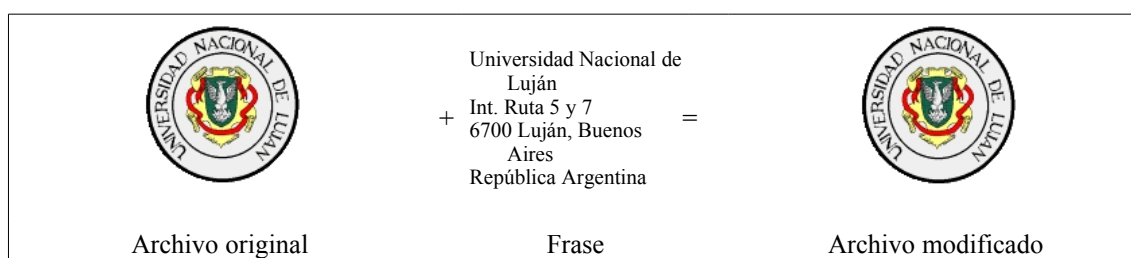
function Decrypt(ciphertext, f) {
    var key = parseInt(document.getElementById("key").value);
    pt = new Array(ciphertext.length);    k=0;
    for(line=0; line<key-1; line++){
        skip=2*(key-line-1);    j=0;
        for(i=line; i<ciphertext.length;){
            pt[i] = ciphertext.charAt(k++);
            if((line==0) || (j%2 == 0)) i+=skip;
            else i+=2*(key-1) - skip;
            j++;
        }
    }
    for(i=line; i<plaintext.length; i+=2*(key-1)) pt[i] = ciphertext.charAt(k++);
    return pt.join("");
}
```

3.3. Esteganografía

La esteganografía es la rama de la criptografía que trata sobre la ocultación de los mensaje para que no se perciba ni siquiera su propia existencia [WIKI_ESTES] [CRIPTO]. En contraste con la criptografía, donde está claro que el mensaje existe pero el contenido se encuentra codificado, los mensajes en la esteganografía se ocultan en sí mismos, por ejemplo, ocultando texto manipulando el tamaño de la letra, el espaciado

entre caracteres, o una muy utilizada, ocultándolo dentro de una imagen. Para esto existen varias aplicaciones que permiten codificar mensajes dentro de imágenes de una manera muy sencilla, donde se recibe como parámetro una imagen cualquiera, una contraseña y la frase que se desea codificar, y basado en ello se genera una imagen de aspecto idéntico con la frase oculta, a la cual hay que aplicarle una función para recuperarla [CRIPTO].

Por ejemplo, a continuación se muestran dos logotipos con el escudo de la Universidad Nacional de Luján. El primero es el archivo original obtenido de la página de internet de la misma. El segundo logo se generó añadiendo los datos postales de la UNLu mediante una aplicación web *cryptimg*²³ que retorna el archivo modificado.



A estos archivos se les aplicó un *diff*, para confirmar la inserción de la frase de manera no legible en modo texto y no visible en modo imagen.

```
pablo@pablo-laptop:~$ diff -a "logoOrigil.jpg" "logoCrypt.jpg"
49c49
< 0L000I@0#+000a000
8000m#00#00~#00~#0000#0
---
> 0L000I@0#+000a000
8000m#00#00~#00~#0000#00; &1-
VWAM5pZXdmESVyOTc2MC1kHPYWCCQgYUTmCCFjQVaWKS9uXJYWBUwgYTZGMZUgHMTMGVqKRw6YYFuKKDQBRpJAD
bnUYQuXWIFBOJ1SLdGGPEgVINSJUB5CWIDUJcNjGcJpTY3XNMDSHAgYTHZyVqTW6WUFuQXLCIIBCJPdWYFVuYI
b3ZAMgEUQWJ01AYZXIYMNMAC1FLJ1UFcMKQ06YGymLOxpETY2XOEgJVQXTHJnDTZWHi50NRaWWO5h; &1-31QE86
MCbaFSc1OZ1fXye9QV5fFB0dKJdfVI2eYec9DRd6MG35LT9aOZfeKL152
pablo@pablo-laptop:~$
```

3.4. Funciones Hash

Una función resumen o hash en inglés [CRIPTO], es un método que se aplica a un documento para garantizar la autenticación del mismo mediante una

23 <http://www.vicio2k3.net/services/cryptimg/>

secuencia de bits de pequeña longitud adjuntado al mensaje en lugar de tener que utilizarlo por completo. Como características fundamentales para un buen uso de esta *Funcion Hash*, se destacan:

- Que el resultado de la función resumen sea de longitud fija, independientemente de la longitud del mensaje.
- Que sea fácil de calcular sobre cualquier mensaje.
- Que sea computacionalmente intratable recuperar el mensaje a partir de la función resumen.
- Que sea computacionalmente intratable lograr generar la función resumen desde un mensaje distinto. De acá deriva una condición implícita que determina que el tamaño del resultado de la función resumen debe ser de al menos 128bits para poder reducir las probabilidades de que dos mensajes aleatorios den el mismo valor resumen.

3.4.1 MD5

MD5²⁴ es un algoritmo de reducción criptográfico ampliamente difundido en la actualidad, diseñado por el profesor Ronald Rivest, en el año 1991, luego de que en el mismo año, Den Boer y Bosselaers publicaran ciertas debilidades en su antecesor, el MD4. En el año 1996, Hans Dobbertin anunció que el MD5 tenía problemas de colisión de hash. En agosto de 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu anunciaron los resultados de un ataque de colisión de hash [CRIPTO] [CollHash] [COL_Stevens] que se consumó en poco más de una hora con un clúster IBM P690. Con las dudas ya planteadas, quedó demostrado que un hash de 128bits era muy pequeño por lo que varios especialistas de seguridad recomiendan sustituir al MD5 por algoritmos alternativos tales como SHA-1 o RIPEMD-160.

Una forma interesante de demostración de colisiones en MD5 es el trabajo presentado por Marc Stevens²⁵, Arjen K. Lenstra²⁶ y Benjamin de Weger²⁷, quienes crearon el 30 de noviembre de 2007 doce documentos [COLL_PRED] con distinto contenido pero el mismo hash con la predicción de quién ganará las elecciones en Estados Unidos en el año 2008.

24 MD5: Acrónimo de Message Digest Algorithm 5, es un algoritmo de reducción criptográfico de 128bits.

25 http://db.cwi.nl/personen/publiek/zoek_show.php4?persnr=2163

26 <http://people.epfl.ch/arjen.lenstra>

27 <http://www.win.tue.nl/~bdeweger/>

3.4.2 SHA

El algoritmo SHA(*Secure Hash Algorithm* o *Algoritmo de Hash Seguro*) [WIKI_SHA] fué desarrollado por la NSA en el año 1993 y evolucionó a versiones posteriores publicadas con los nombres SHA-1 (ambas de 160 bits) y SHA-2 (esta última con variantes en el rango de salida, llamadas SHA-224, SHA-256, SHA-384, y SHA-512) [CRIPTO]. La importancia que representa el hash de salida y la característica fundamental de que sea teóricamente única, es imprescindible para garantizar que no se encuentren mensajes con firmas iguales. Por ello el incremento de longitud de las funciones SHA de hasta 512 bits, aunque expertos de la talla de Bruce Schneier²⁸ y Kevin Henry[COLL-Alter] declaren ir buscando funciones hash que permitan el reemplazo de SHA-1, analizando alternativas como Tiger o Whirlpool.

Como se mencionó anteriormente, el SHA-1 (al igual que el MD5) son algoritmos de una sola vía y por ello no es posible obtener el mensaje original a partir de su hash al utilizar fuerza bruta. Es más, como son funciones resumen, es matemáticamente probable que un único hash pueda ser generado por infinitos documentos, aunque éstos no sean fácilmente hallables. Hay que tener especial atención con algunos sitios de internet²⁹ que garantizan decodificar estos algoritmos cuando en realidad lo que hacen es acopiar frases y codificaciones de usuarios descuidados en una base de datos pública.

3.5. Criptografía Simétrica

La criptografía simétrica es un método criptográfico que utiliza una única clave para cifrar y descifrar mensajes [CRIPTO]. El primer paso para poder establecer una comunicación entre ambas partes es que se pongan de acuerdo de antemano sobre la contraseña a utilizar, y es allí donde se encuentra el mayor inconveniente³⁰. Una vez realizado el traspaso de claves, la comunicación se establece en forma segura.

El problema radica en el canal que se utilizará para enviar dichas claves ya

28 Bruce Schneier: Criptografo, experto en seguridad informática, autor de varios libros de criptografía, jefe tecnológico de Counterpane Internet Security, creador y codiseñador de varios algoritmos como Blowfish, Twofish, MacGuffin, Solitaire, Yarrow y Fortuna.

29 <http://md5.rednoize.com/> : Sitio de recolección de frases y resultados de funciones resumen MD5 y SHA1 (No probar con contraseñas propias).

30 La realidad es que existen pocos canales seguros, ya que los servicios de información mundiales, las escuchas telefónicas y los métodos de interceptación son moneda corriente hoy en día, por lo cual habría que encontrarse personalmente para intercambiar dicha información.

que sería mucho más fácil para el atacante interceptar ese mensaje.

3.5.1 Ejemplo de sistema simétrico

Un buen ejemplo de un sistema simétrico es el empleado por los alemanes durante la segunda guerra mundial llamado *Enigma*, en donde las claves se distribuían a diario en libros de códigos. Cada día el operador de radio consultaba su copia del libro de códigos en busca de la clave del día y con ésta eran cifradas y descifradas las transmisiones de radio de todo el día. Como contraparte, los ingleses utilizaban unas computadoras de uso general llamadas *Bombes*, para encontrar la clave del día. Algunas veces la clave era encontrada en un par de horas desde que fuera puesta en uso y así podían decodificar el resto de las conversaciones, otros días esto era imposible ya que la clave no había podido ser descubierta.

3.5.2 DES

El algoritmo DES (*Data Encryption Standard*) [WIKI_3DES] está basado en el algoritmo Lucifer (de Horst Feistel), desarrollado por IBM a principios de los setenta y luego adoptado como estándar por el gobierno de los Estados Unidos para comunicaciones no clasificadas. En el año 1975 fué publicado en el Registro Federal donde se solicitaron comentarios del público. Un año más tarde empezaron a llover críticas desde varios sectores (donde se encontraban entre otros Martin Hellman y Whitfield Diffie), resaltando como falencia la corta longitud de la clave (La NSA había aconsejado reducir la clave de 64 a 56 bits). Según varios autores, se sospechó que el algoritmo había sido debilitado adrede por la agencia de inteligencia para que ellos (y solo ellos) pudieran leer los mensajes cifrados de fácil manera.

Muchos de los usuarios de DES, al verse amenazados por la aplicación exitosa de fuerza bruta, comenzaron a utilizar 3DES, que consiste en aplicar DES tres veces consecutivas con diferentes claves [CRIPTO] . Seguro por algún momento pero con costos computacionales muy altos. Empezaron a aparecer alternativas como DES-X o GDES, pero su muerte ya estaba anunciada. En el 2001, tras un concurso internacional, el NIST (*National Institute for Standards and Technology*) escogió un nuevo algoritmo, el AES (*Advanced Encryption Standard*) .

3.5.3 AES (o algoritmo de Rijndael)

En el año 2001, el algoritmo de Rijndael [WIKI_AES] fue anunciado como nuevo estándar avanzado de cifrado para empleo de aplicaciones criptográficas no militares, potente, eficiente y fácil de utilizar [CRIPTO]. Creado por los Belgas Joan Daemen y Vincent Rijmen, es un sistema de cifrado por bloques, diseñado para manejar longitudes de clave variable comprendidas entre 128 y 256 bits. Según sus autores, y basado en la estructura de su diseño, el método más eficiente conocido hasta la fecha para recuperar la clave a partir de un par *texto cifrado/texto claro*, es la búsqueda por *fuerza bruta*, por lo que se considera en la actualidad como uno de los métodos más seguros.

3.6. Firma Digital

La firma digital es un método criptográfico que se aplica a un documento, el cuál permite asegurar la identidad del firmante y la integridad del mensaje a partir de una huella digital o secuencia de bits añadida a la pieza original de información. Este método presenta una analogía directa con la firma autógrafa, por ende debe cumplir con tres propiedades fundamentales:

- Estar ligada directamente a un mensaje y no ser válida para un documento diferente.
- Solo puede ser generada por su emisor original, de la misma manera que una firma autógrafa se encuentra vinculada con una única persona.
- Es públicamente verificable, cualquier persona podría comprobar su identidad de manera sencilla.

Como se muestra en la figura siguiente, la forma más sencilla de generar una firma digital consiste en emplear una combinación de cifrado asimétrico y funciones resumen.

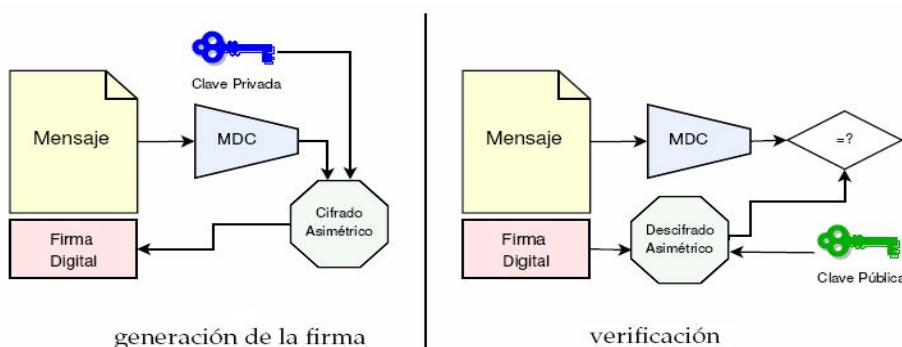


Figura 3.1 – Generación y verificación de firma digital.

3.7. Criptografía Asimétrica

La criptografía Asimétrica [CRIPTO] [WIKI_ASIM] es el método criptográfico que utiliza un par de claves (una pública y otra privada) para la transmisión de mensajes. Ambas claves son generadas en el mismo momento, la clave pública se entrega a las terceras partes, y la clave privada se deberá guardar de modo que nadie tenga acceso a ella. Estos sistemas fueron creados para evitar el problema de intercambio de claves que contenían los sistemas criptográficos simétricos. El método de comunicación se realiza a partir de encriptar un mensaje con la clave pública del destinatario, de manera que solo él, con la clave privada, pueda descifrar el envío. Ya que las claves públicas son de dominio corriente, conseguir una copia de la misma no representa problema alguno para este sistema. Este método se utiliza en el caso que se desee proteger la información desde un emisor (el que conoce la clave pública) a un receptor (que obviamente posee su clave privada). La otra forma de utilizar algoritmos asimétricos es para la autenticación de mensajes. Mediante el uso de firmas digitales, el emisor (en este caso el que posee ambas partes, la clave pública y la privada), tiene que generar un resumen del mensaje y codificarlo con su clave privada. El receptor (parte que desea comprobar la autenticidad del mensaje y conocedor de la clave pública del emisor) puede ahora descifrar y comparar el criptograma. Si coinciden, el mensaje es auténtico ya que el único que posee la clave privada para codificar el mismo es el emisor.

3.7.1 Funcionamiento

Los sistemas de cifrado mediante pares de claves pública/privada se basan en el aprovechamiento de algunas propiedades particulares de las matemáticas que otorgan, por ejemplo, los números primos. Como se sabe, es fácil multiplicar dos números primos para obtener uno compuesto, pero extremadamente difícil factorizar uno compuesto en sus componentes primos. Pero si se conoce uno de los factores, es muy fácil hallar el componente que falta. El cifrado de pares de claves utiliza algo parecido para poder computar el inverso de la función.

3.7.2 RSA

El sistema criptográfico con clave pública RSA [CRIPTO] [WIKI_RSA] es

un algoritmo criptográfico asincrónico descrito en 1977 por Ron Rivest, Adi Shamir y Len Adleman del MIT, pero hecho público recién en 1997 ya que el mismo fue confidencial hasta entonces. Este algoritmo se basa en la dificultad de factorizar grandes números. Las claves pública y privada se obtienen a partir de números primos grandes. Aunque el algoritmo RSA es bastante seguro conceptualmente, existen algunos puntos que se deben tener en cuenta, como ser el tamaño de la clave, que no debería ser menor a 1024 bits; y no firmar el mensaje después de codificarlo, ya que existen algoritmos que permiten manipular con éxito mensajes primero codificados y luego firmados.

Existen otros algoritmos de criptografía asimétrica como el Cifrado ElGamal, el DSA o Digital Signature Algorithm, el Algoritmo de Rabin, el protocolo de Diffie-Hellman o el CCE o Criptografía de Curva Elíptica, aunque para profundizar en estas variantes se recomienda extender la lectura en libros dedicados³¹.

3.8. PGP

PGP (o Pretty Good Privacy) [WIKI_PGP] es un programa que permite el cifrado de datos, mensajes o archivos, diseñado en sus orígenes para asegurar el correo electrónico, aunque luego fue incorporando funcionalidades tales como el cifrado de archivos con claves simétricas, creación de VPNs, discos virtuales cifrados y borrado seguro de datos. Este proyecto fue iniciado a principios de los 90 por Philip R. Zimmermann para cubrir la ausencia de herramientas sencillas, potentes y baratas que vincularan al usuario común con el uso de métodos criptográficos serios [CRIPTO.ES].

La criptografía de clave pública nació en 1977, con el descubrimiento de los sistemas RSA y Diffie-Hellman. En esos momentos la criptografía era utilizada solo por organismos militares y unos pocos comerciales ya que la NSA restringía fuertemente su uso por la lucha contra el espionaje electrónico. Estrictamente hablando de su diseño, es un sistema híbrido, en el que la clave pública cifra una clave simétrica que a su vez cifra el mensaje.

Como se mencionó anteriormente, PGP utiliza dos formas diferentes de clave pública: RSA (con función MD5) y Diffie-Hellman (con una función hash SHA-1), con una longitud de clave de hasta 2048 bits y 4096 bits respectivamente. Los algoritmos de clave simétrica que puede utilizar son: CAST de manera predeterminada, pero se puede optar por IDEA o 3DES [CRIPTO]. Las claves públicas se obtienen

31 Criptografía y Seguridad en Computadores, 4ª Edición. Versión 0.7.8, de Manuel J. Lucena López.

generalmente de servidores de claves en donde se dan de alta las mismas junto con algunos datos como para que el resto de los usuarios puedan llegar a recuperarlas.

3.8.1 Historia

En tiempos donde el criptoanálisis y los sistemas de cifrado eran controlados de cerca por la NSA, y la exportación de programas que utilizaban esta tecnología se encontraban prohibidos por Estados Unidos (se encontraban al nivel de exportar armas sin licencia con penas de hasta 10 años de cárcel y multas de un millón de dólares) y además justo cuando el congreso se encontraba dando los primeros pasos para sancionar legalmente la obligación de que los mismos tuviesen “puertas traseras” que permitiesen al gobierno leer los mensajes cifrados; un desconocido ingeniero de software llamado Philip Zimmermann [WIKI_PHIL] [ZIMMERMANN] implementó el sistema RSA junto con un algoritmo simétrico propio (Bass-O-Matic) y la función resumen MD4 para crear la versión 1.0³² del PGP.

Con una versión de altas prestaciones y diseñada para el uso general, Philip distribuyó varias copias del PGP entre sus amigos, algunas de ellas se diseminaron entre tableros electrónicos BBS³³, y de algún modo ellas salieron de Estados Unidos para que el gobierno de ese país tuviera fundamentos para acusar a Zimmermann de violar las leyes de exportación ITAR³⁴ [WIKI_ITAR]. Conocida es la historia de un ingeniero noruego llamado Stale Schumacher, quien descubrió que lo que sí se podía sacar de Estados Unidos era el código fuente del programa escrito en liberos (consecuencia de las leyes que protegen la libertad de prensa), así que compró los mismos en su país y los recompiló luego de aplicarle un OCR. Esta versión recompilada se denomina como “versión internacional” y es reconocida por una letra *i* luego del número de versión.

A partir de las versiones 2.6.2 ya se podían ejecutar en los MS-DOS de la época (primeras versiones masivas), luego fueron creadas múltiples interfaces para Windows 3.1 y luego de su versión 5.0 ya se podían ejecutar en los Windows 95/98; siempre de la mano de Philip y su empresa PGP Inc.

32 Luego esta versión evolucionaría, incorporando un algoritmo simétrico más fuerte (IDEA), una función resumen más moderna (MD5) y un algoritmo de compresión diferente (ZIP en lugar de LZHuf).

33 Bbs o **Bulletin Board System** era un medio de intercambio electrónico de mensajes utilizado en la época quedando obsoleto con el advenimiento de la **World Wide Web**.

34 ITAR: International Traffic in Arms Regulations, quien controla para el gobierno de Estados Unidos la exportación e importación de artículos relacionados con defensa.

Por otra parte, la costumbre de hacer público el código fuente ha contribuido a la aparición de versiones alternativas open source como OpenPGP, desarrollado para entornos Unix/Linux, Mac y Windows [CARVANTES] [CRIPTO.ES].

3.8.2 Encriptación

Dentro de las operaciones que se pueden realizar se encuentra la encriptación. Cuando un usuario desea enviar un documento encriptado con PGP, lo primero que debe hacer es obtener la clave pública del destinatario (como se mencionó en el apartado anterior, esto se puede ubicar a partir de servidores públicos en internet). Dicha llave se utilizará para encriptar el documento, y puesto que la información solo puede descryptarse con la otra llave de la misma pareja, se puede garantizar que solo el destinatario, único poseedor de la llave privada, podrá leerla en texto claro. Como ejemplo de su uso podemos ver el siguiente texto de referencia:

```
La Universidad de Lepe acaba de publicar un estudio en el que se demuestra que utilizar
Windows prolonga la esperanza de vida. Investigaron la salud y la longevidad de dos
millones de personas. La mitad eran europeos que cotidianamente utilizan ordenadores con
el sistema Windows. El otro millón eran africanos que jamás han usado un ordenador. Los
que emplean Windows viven más años y tienen mejor salud que los otros.
```

A partir del texto claro dado como ejemplo, se puede observar la codificación una vez aplicado el PGP.

```
-----BEGIN PGP MESSAGE-----
Version: PGPfreeware 5.5.3i for non-commercial use <http://www.pgpi.com>

hQCMA6arQCE08evfAQP8CbOM05vfadHRJo8kwNnM5HHG7C30dWmNxu5CPA+m+LQY
bJBYFjMQqNK/NPjvkdjqU2mCWPhJa2nSH+X5xuneLfXnXAxzwvjP7w4a/NFILgyo
VDQzMQScheakDWxScApi4pi/CxfFLVKXOPunFHssJLy5ZcHFWLQfABNuRTI/Gzil
ARuCla0XDs32A/wvChx2SYlkZnQySzjYehcsRISHXlzkX5H7LoUSgaHW6xW/4tYx
nN8N3+VBTyZ7Nxa78dEa3AShWRW3F1IvOj+zv8x+15OwHZsXYotVY120b2EVUMF5
bzzkQsFBFt/m4T0NdG/sFThh7uOGxedBSiV4PrdLO6Wb192XeLIYI1OHnWAEAkqd
GOr5lZj+TfJTVLHdVfK+7WpTa/vgaYp1w3pASGNCdnThFlQMn5VCKSgD8d6u5KNX
Y6l5YRbaXhatqH1io3XLUgzs6MlMskJ/VFCTQLvpxxU74VVkcZ0ebWO/OaX/cDVt
CbnFphtoAUoe70M0AkGhsxZz1wtSviE8Z/4dJOG8x+maNhPTGw8KBHvH/aDy
=uA7n
-----END PGP MESSAGE-----
```

Entre los tags claramente demarcados se encuentra el texto encriptado que generalmente ocupa un número menor de bytes que el escrito original ya que al mismo

se le aplica además un método de compresión. Es posible que la información encriptada sea enviada a varios destinatarios por lo que dicho mensaje encriptado ocupará algunos bytes de más por el encabezado extendido.

3.8.3 Firma Digital

Otra aplicación del PGP es utilizarlo como método para asegurar que el documento procede de la persona que dice remitirlo, y a su vez certificar que el mismo no ha sido modificado por terceros. La información transmitida es comprimida en una primera instancia y luego encriptada con la clave privada del remitente. Junto con la misma es añadido el documento original. Cualquiera puede ver la firma digital adjunta, que fácilmente podría ser desencriptada con la clave pública del remitente para validar su contenido. Esto garantiza que el documento original fue firmado con la clave privada en posesión de una única persona (el remitente). A continuación se presenta un ejemplo de lo mencionado en este último párrafo [CRIPTO.ES] [CRIPTO].

-----BEGIN PGP SIGNED MESSAGE-----

La Universidad de Lepe acaba de publicar un estudio en el que se demuestra que utilizar Windows prolonga la esperanza de vida. Investigaron la salud y la longevidad de dos millones de personas. La mitad eran europeos que cotidianamente utilizan ordenadores con el sistema Windows. El otro millón eran africanos que jamás han usado un ordenador. Los que emplean Windows viven más años y tienen mejor salud que los otros.

-----BEGIN PGP SIGNATURE-----

Version: PGPfreeware 5.5.3i for non-commercial use <<http://www.pgpi.com>>

iQCVAwUBNWbZpaarQCE08evfAQHL6wQAsNPP+HamC8i9HVmwOWgXEy8d69Rcsfi
t4U8+32lMYBt+A3o4sycE06b8dZJ2KuosjYl1T0KJba/oolyeUVBICEEXwAUK7/D
cUtEzE6Gf8vHfmSn4aE0Ta68RqUYe4aq1lo/yIMbkmxMh5A/7/e9qzN0lmwipqQA
Sz0xh5vzlhs=
=fkrF
-----END PGP SIGNATURE-----

3.8.4 Firma con Encriptación:

El PGP puede aplicar ambas operaciones firmando en primer lugar el documento (como se menciona en el punto 2.3.8.3) y al conjunto resultante se le aplica el proceso de encriptación con clave pública del destinatario (como se indica en el punto

2.3.8.2). Para el inverso, el receptor deberá utilizar su llave privada para deshacer la encriptación y luego utilizar la llave pública para verificar la firma digital. Como último ejemplo se presenta a continuación el mismo texto, ahora firmado y encriptado.

```
-----BEGIN PGP MESSAGE-----
Version: PGPfreeware 5.5.3i for non-commercial use <http://www.pgpi.com>

hQCMA6arQCE08evfAQQAgaTgXXYniEUdpOkPba6Qm3PVF5g9oraht9FkyooDEND1
JH2/gvRENxcGipUsDjrQgXZNkptBwmVHzK/6AsfeLsWaXDZ6+WkyhCFC6jNLXno0
Mq4Esi7urNXY6f/uBpg3ToROlJV/88zM+dCl1LdGZ2MwzTjjEkKnBac4ChUgHJml
Ae0BlHqdvxHGd9hileZlimF7LgBRs2+zANJhjarAe1c5ytm8v96yK8g+XK9oyDnC
dmW/tTz5q3j8/nSIJAWApdb/zHnJKLclbI8v1TsFQFL36dNDBwLxQ40iYkjB5Ol+
P7XInWltUWE0QiuHKfoTeFFcm1w/PzHxFDE356x3e1ttzo+SOss0jo0ANLxRzKHg
Dd6hR9ujfI6a3VgCO7viRYvfvJgm53Act92r2awJsHCV7KyRbg4Y1aI2NW+HQwSO
qQSCXgco7g+LdYombHb2FWboHCsmLjkT/9zXBjPpfleugHwghPcHdZc1Hbo9k8Mo
55mSZzldh+9riGR3uodq08iwGI0SMYfpDRdwoboaq4BjQXuGUU+9nbG2fjM6dVng
IGhg1cXz39TpMiWW4eIjmZ3B5Lz+799q4OZhIv0mns2lTA1ECIadOwYou/h+NJD0
hMPKPFv86MvqRdPbCGr5amssv723aVSCBaewwbNHaZM01QU9xKRVoIUGVuTOxp/u
PxZKdIGSKVwEzKMqZNrxWMxedCTb3bRRY4iAX4YB58k8w1BpFI7eiaVpnqHv18cd
JVG9/7eDC+Tmqslzzzt0iojbcHtxUrbq/biQUQDVgUHwmcIJmxOzyiQqQSVRBU5E
IhsJxJy6liujlq+3s5G8
=UTPN
-----END PGP MESSAGE-----
```

3.9. GnuPG

GPG [GnuPG] es una herramienta de cifrado y firmas digitales que utiliza el estándar IETF³⁵ denominado OpenPGP (RCF 4880) de similares características que el PGP pero con la gran diferencia que es Software Libre licenciado bajo GPL ya que no utiliza el algoritmo patentado IDEA.

Este software [WIKI_GPG] fué inicialmente desarrollado por Werner Koch, quien lanzó la versión 1.0 el 7 de septiembre de 1999, consiguiendo luego el apoyo del gobierno alemán para financiar la documentación y su versión para Microsoft Windows en el año 2000. Su última versión, totalmente estable y calificada para ambientes de producción, actualmente se incluye en OpenBSD, FreeBSD, NetBSD y GNU/Linux, en este último integrado también en el Kmail y el Evolution. Además existe una versión compilada para plataformas Mac OS X (MacGPG) y Windows.

³⁵ Internet Engineering Task Force, o Grupo de Trabajo en Ingeniería de Internet, es una organización internacional abierta de normalización, que tiene como objetivo contribuir a la ingeniería de Internet, creado en 1986 en Estados Unidos.

Al igual que el Pgp, Gpg cifra los mensajes usando pares de claves asimétricas, una privada y otra pública depositada en servidores de claves. En lugar del algoritmo IDEA³⁶, presente en Pgp, Gpg utiliza algoritmos no patentados como ElGamal, CAST5, 3DES, AES y Blowfish.

3.10. Conclusión

Para cerrar este capítulo, es importante tomar conciencia que las herramientas para mantener la seguridad e integridad de los datos están dadas y que se encuentra disponible un amplio abanico de alternativas como para que a partir de ahora sea responsabilidad propia el tratamiento de la información en forma segura y así establecer una barrera mas en el correcto manejo de los datos.

³⁶ En realidad es posible utilizar el algoritmo IDEA descargando un plugin extra, el cual puede llegar a requerir licencia en países donde está patentado.

4. Sistemas de monitoreo

4.1. Técnicas de Detección de Sniffers

La detección de sniffers en la red es una de las múltiples tareas que tiene el administrador de seguridad para que la estabilidad de la misma no se vea comprometida. Para poder lograr esto, es necesario en primer lugar, conocer como funciona este tipo de aplicaciones, ya que no existe una receta definitiva para identificar las máquinas que operan en modo promiscuo y se debe utilizar el sentido común para identificar los posibles falsos positivos y sobre todo poder encontrar los falsos negativos [RYSP03] [HSRR00].

4.1.1 Qué es y cómo funciona un Sniffer

Un Sniffer es un dispositivo (escapa a una aplicación ya que existen Sniffers de hardware), que permite tener acceso a tráfico al que en realidad no debería, ya que no estaba dirigido expresamente a él ni al broadcast. Ahora, ésta sería una definición muy amplia, pero en realidad la detección de sniffers en una red apunta a que el administrador ubique posibles amenazas de seguridad, originadas generalmente en equipos donde existen placas de red escuchando en modo promiscuo. Por ello en el futuro se apuntará a encontrar dichas placas de red, ya que las mismas serían el verdadero problema [PASNIFF].

En condiciones normales, los paquetes recibidos son aceptados o rechazados a nivel de hardware por la placa de red según la MAC ADDRESS destino que se especifica en el *frame Ethernet*. Solo en el caso de que esa MAC ADDRESS sea coincidente a la de la placa propia o la de broadcast, el paquete será aceptado y procesado; en caso contrario se rechaza (ignora). Pero cuando una placa de red se encuentra funcionando en modo promiscuo, todos los paquetes son aceptados y procesados, por lo que pasan directamente al kernel del sistema operativo quien ahora será el encargado de filtrar y decidir que paquetes serán descartados y cuales serán procesados por otras aplicaciones. La gran diferencia para encontrar las placas de red que están en modo promiscuo es el método para buscarlas, ya que es mucho más fácil (pero lamentablemente menos usual) reconocerla en forma local que en forma remota, proceso que resulta muchas veces complejo, ambicioso, y prácticamente imposible [WIKISNIF] [CHRE].

4.1.2 Técnicas locales de detección:

Esta es la forma más trivial de localizar un sniffer. Si se dispone de acceso local a la máquina, en los sistemas operativos de la familia Unix se dispone de una utilidad comúnmente usada pero pocas veces con atención. El comando *ifconfig* informa, además del estado de las interfaces, si la misma se encuentra en modo promiscuo o no. Una buena práctica es verificar el estado de las interfaces con este comando periódicamente, y una manera de automatizarlo sería agregando dicha tarea en el *crontab* para que, en caso de descubrir que ha cambiado a modo promiscuo, envíe un alerta al administrador de la red (también podría deshabilitar la misma inmediatamente). Si se desea una alternativa al *ifconfig*, el programa *cmp* (siglas de Check Promiscuous Mode) realiza una tarea semejante mediante el uso de llamadas *ioctl*. Es evidente que la no detección temprana de una posible violación de seguridad puede hacer que el atacante disponga de tiempo como para sustituir el ejecutable *ifconfig* o *cmp* compilando uno de similares características pero que no informe el cambio de funcionamiento de la interfaz de red; o de la misma manera, deshabilitar la tarea programada en el *crontab*. Igualmente la integridad de los ficheros ejecutables es fácilmente comprobable utilizando funciones resumen como MD5 o SHA-1 (como se verá en capítulos posteriores).

```
root@pablo-laptop:~# ifconfig eth0
eth0      Link encap:Ethernet  direcciónHW 00:16:D3:9E:DD:64
          inet addr:10.218.106.xx  Difusión:10.218.106.255  Mask:255.255.255.0
          inet6 addr: fe80::216:d3ff:fe9e:dd64/64 Scope:Link
          UP DIFUSIÓN CORRIENDO PROMISC MULTICAST MTU:1500 Metric:1
          RX packets:25255 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12229 errors:0 dropped:0 overruns:0 carrier:0
          colisiones:0 txqueuelen:1000
          RX bytes:7284325 (6.9 MB)  TX bytes:2076777 (1.9 MB)
          Interrupt:17 Base address:0xe000
root@pablo-laptop:~#
```

4.1.3 Técnicas de detección remota:

Existen numerosas técnicas de detección de *sniffers* para el segmento de red, aunque hay que tener en claro que las mismas tienen ciertas limitaciones porque en general hay posibilidades de que igualmente existan *sniffers* en la red sin ser detectados, o también, máquinas completamente inocentes que sean detectadas como falsos positivos. Estas técnicas se pueden clasificar en las que son dependientes del sistema

operativo, y las que no [CHRE].

4.1.3.1 Técnicas dependientes del sistema operativo.

Como su nombre lo indica, estas técnicas aprovechan algún fallo del sistema operativo (o de su subsistema encargado del TCP/IP) para reconocer que una placa de red se encuentra en modo promiscuo. Con un muy buen rendimiento cuando se exploran máquinas donde coincide exactamente la versión del sistema operativo utilizado, la gran desventaja es la cantidad de falsos negativos que genera.

Un ejemplo es aprovecharse de los kernel antiguos de sistemas linux, donde los paquetes se analizan exclusivamente según los datos a nivel IP. Si se crea un paquete con una MAC ADDRESS de destino que no exista en el segmento, y encapsula un paquete del tipo ICMP echo reply (ping), con dirección destino la IP de la máquina a analizar. Las computadoras que se encuentren en modo no promiscuo rechazarán directamente los paquetes que tienen como MAC ADDRESS un destino que no es la suya, las placas que se encuentren *sniffeando* pasarán el paquete al *kernel*, quién analizará, como se dijo, el mismo a nivel IP, y desde ese punto de vista éste es un ping normal, contestará el ping, y así revelará su estado [HSRR00].

4.1.3.2 Técnicas no dependientes del sistema operativo.

En general, estas técnicas son menos fiables ya que suelen basarse en suposiciones sobre el comportamiento de determinados *sniffers*. A continuación se detallan diversas técnicas de esta metodología.

4.1.3.2.1 Test de DNS.

Los test de DNS se basan en la suposición de que los *sniffer* transforman las direcciones IP en sus correspondientes nombres para facilitar la lectura del tráfico en su presentación. Si este es el caso, se podría tratar de "engañar" al sniffer con paquetes falsos que contengan una IP nueva. Todas las máquinas, al ver que esta IP no les pertenece, descartarán dichos paquetes. Pero el sniffer, queriendo convertir la IP en un nombre, y al no disponer el mismo en su cache local, hará una consulta de DNS y de esta forma revelará su situación.

4.1.3.2.2 Test de latencia.

En este caso, el método se aprovecha de la condición de que en modo

promiscuo, los paquetes se rechazan o aceptan a nivel de aplicación (ya que todos pasan al kernel como propios). En forma normal esto ocurriría a nivel de hardware lo que implicaría tiempos menores de respuesta si se analizan las estadísticas de tráfico. Es por eso que lo único que hay que hacer es analizar dichos tiempos. Aunque lamentablemente es posible evitar ser detectado como sniffer si se monitorea en busca de una drástica inundación de tráfico en la red, de manera que en este caso se pudiera cambiar la placa de red de modo promiscuo a modo normal; técnica de evasión ya implementada en varios de los sniffers actuales [HSRR00].

4.1.3.2.3 Test ARP.

Se puede enviar una petición ARP al objetivo con toda la información necesaria pero con la MAC ADDRESS errónea. Una máquina que no esté en modo promiscuo nunca verá ese paquete, puesto que no era el destino original, por lo tanto no contestará. Si en cambio, la placa de red se encuentra en modo promiscuo, la petición ARP será considerada y el kernel la procesará contestando y quedando evidenciada.

4.1.4 La lucha continua

En la actualidad existen varias herramientas de detección de sniffers como *Ettercap*, *Neped*, *Antisniff* y *Sentinel*; este último con menos desarrollo a la fecha pero al ser un proyecto público y abierto, de muy buena proyección. Esta continua lucha entre la evolución de los sniffer y los anti-sniffers se debe seguir de cerca ya que por ahora la balanza se encuentra inclinada en contra del administrador de seguridad por lo que se deben aplicar políticas alternativas para minimizar el impacto de los sniffers como la segmentación de redes por ejemplo VLANs (físicamente o por medio de switchs inteligentes) o la utilización de tráfico cifrado mediante el uso de VPNs.

4.2. Sistema de Detección de intrusos

Un Sistema de Detección de Intrusos o IDS es un programa utilizado para detectar accesos no autorizados a una red, técnica relativamente nueva que se agrega a los métodos ya conocidos de defensa, básicamente para recolectar información utilizando pluggins como sensores que buscan patrones o firmas de ataques conocidos analizando el tráfico de una red. Estos sistemas de detección pueden almacenar dichas señales y dar avisos de alerta al administrador de la red, o actuar en forma reactiva por

ejemplo, reprogramando el firewall de la red o intercambiando información con otro componente de la seguridad del sistema [HSRR00] .

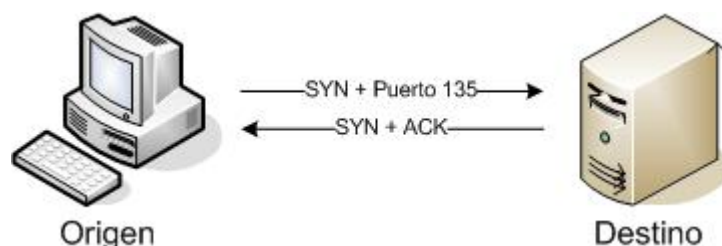
4.3. Portsentry

El primer paso para poder concretar un ataque es recopilar cuanta información sea posible acerca de la víctima. Dicha información ronda principalmente los puertos abiertos de la máquina, a partir de ahí conocer que servicios dispone, y luego obtener, si es posible, las versiones de software que utiliza. Estos datos son suficientes para que un ataque sea exitoso, tan sólo con encontrar una versión desactualizada de uno de los programas instalados, y explotar a partir de ahí alguna de las vulnerabilidades conocidas.

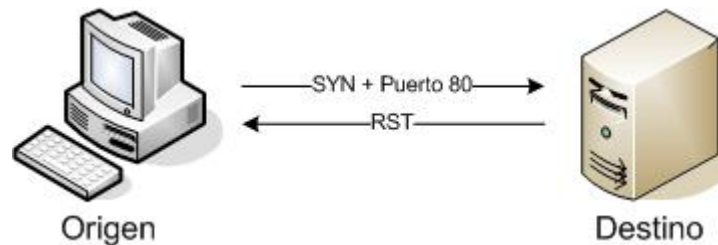
La manera más común de que un atacante obtenga información, es realizar un barrido de puertos mediante algún programa dedicado, como por ejemplo *Nmap* [WWWNMAP]. *Nmap* es una herramienta de escaneo de puertos creado para analizar grandes redes a una velocidad relativamente rápida para obtener, entre otras cosas, las IPs de los equipos que se encuentran conectados en la red, los servicios, versiones, nombres de programas, sistemas operativos, filtros de paquetes de firewalls, y otras muchas características más.

Dichos puertos se pueden encontrar en diferentes estados que se describen a continuación [GNU4ALL]:

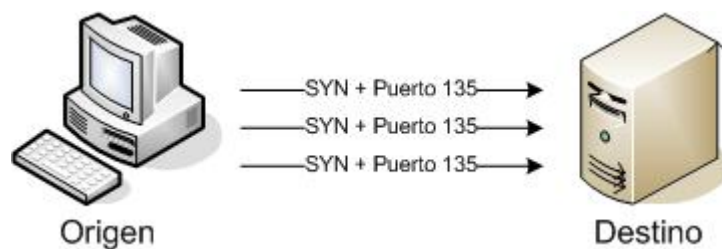
- **Puerto Abierto:** Un puerto se encuentra abierto cuando al enviar un SYN para intentar establecer una conexión, el destino responde exitosamente con un SYN +ACK. Es este estado el que generalmente los atacantes buscan para explotar vulnerabilidades, ya que se sabe que un servicio se encuentra detrás esperando conexiones.



- **Puerto Cerrado:** De manera inversa, un puerto se encuentra cerrado cuando como respuesta al SYN se recibe un RST (o destino inalcanzable en el caso de un test UDP), lo que significa que no existe servicio esperando conexión por este puerto.



- **Puerto Filtrado:** En el caso de los puertos filtrados o bloqueados, a cualquier envío de SYN (o intento de establecer una conexión), no se recibe nada. Muy probablemente se deba a que el destino tiene un *firewall* que elimina el flujo normal de los paquetes.



4.3.1 Funcionamiento típico:

La detección de un barrido de puertos es muy fácil: Se generan muchas conexiones simultáneas a una gran cantidad de puertos, originadas por la misma dirección IP, según las respuestas de dichas conexiones se imprime el resultado del test de barrido. Con el correr de los años, la técnica que utilizan los escaners ha ido evolucionando mediante el cambio de estrategias (aumentando los tiempos entre pruebas, hasta desarrollar un barrido silencioso con SYN, FIN, Xmas, Null, UDP, paquetes fragmentados y barridos paralelos de diferentes tipos) pero siempre bajo el mismo principio [SECFOC].

Ahora que se conocen las posibilidades de los escaners de redes, y como contrarrestar los intentos de ataques, se va a describir una de las herramientas más utilizadas para este fin. Portsentry es una aplicación desarrollada por Psionic que permite detectar escaneos de puertos, incluidos de semiconexión, y actuar en

consecuencia. Su misión radica en escuchar los puertos que según el administrador deberían permanecer inactivos, y en caso de encontrar una conexión entrante a uno de ellos, se procederá marcando este evento en el log del sistema y alternativamente bloqueando la comunicación con la dirección IP identificada o hacer correr algún comando externo.

4.3.2 Configuración de Portsentry

Antes de diferenciar las formas de configuración del Portsentry se debe recordar que el mismo puede ser utilizado en combinación con otras herramientas para transformarlo de un *anti-escaner* pasivo (escucha, loguea, pero no actúa en consecuencia) o en un IDS activo (si se añaden reglas de *Iptables* o, se modifican las reglas de ruteo, o por ejemplo, se escriben nuevas IPs en el archivo *host.deny*) [GWOLF] [ALBA].

4.3.2.1 Modo Clásico o simple:

En este modo se debe especificar en el archivo de configuración una lista de puertos donde Portsentry se encargará de abrirlos y dejarlos a la escucha, esperando que algún atacante los escanee. Se debe tener en cuenta que no es conveniente poner al programa en escucha de puertos que se utilizan normalmente ya que seguro se generarían muchos falsos positivos (si se encuentra en modo pasivo) y hasta puede llegar a cortar una conexión "buena" si se combina con algún tipo de regla de firewall.

4.3.2.2 Modo Avanzado o stealth:

En el *modo stealth*, lo que se hace es especificar un número de puerto para que Portsentry escuche todo lo que está por debajo de él, obviando los ports a los que se dé un uso real (esto lo logra mediante el conocimiento de los servicios propios). Hay que tener en cuenta que existe una lista de excepciones (ADVANCED_EXCLUDE_TCP y ADVANCED_EXCLUDE_UDP) para el caso de tener que excluir puertos por ejemplo, en los cuales se debe actuar sobre redes con máquinas MS Windows donde se tendrán que apartar los puertos de los que NetBIOS hace uso como el TCP 113, TCP 139, UDP 520, UDP 138, UDP 137 y UDP 67.

4.3.3 Ejemplo de configuración:

Sea cuál fuere el método en el que *Portsentry* sea ejecutado, es fundamental

que tengan conectividad desde afuera si el mismo está detrás de un router, ya que el *anti-escaner* no escuchará nada si el router o firewall filtran el intento de conexión antes. Una buena manera de configurar un sistema con *Port Sentry* es, por ejemplo, servir *ssh* en un puerto diferente al 22, por ende este puerto se encuentra en desuso para el sistema, se abre el *tcp* a través del router y se configura el IDS de la siguiente forma:

```
#####
# Advanced Stealth Scan Detection Options #
#####
ADVANCED_PORTS_TCP="1024"
ADVANCED_PORTS_UDP="1024"

#####
# Ignore Options #
#####
# 0 > No hace nada, solo lo notifica
# 1 > Bloquea las conexiones (agrega las IPs en el archivo hosts.deny)
# 2 > Ejecuta un comando externo a PortSentry.
BLOCK_UDP="2"
BLOCK_TCP="2"

#####
# External Command#
#####
KILL_RUN_CMD="/sbin/iptables -A INPUT -s $TARGET -j DROP"

#####
# Dropping Routes:#
#####
KILL_ROUTE="/sbin/route add -host $TARGET$ reject"

#####
# TCP Wrappers#
#####
KILL_HOSTS_DENY="ALL: $TARGET$ : DENY"

#####
# Scan trigger value#
#####
SCAN_TRIGGER="0"
```

De esta manera, el puerto 22 estaría “engañosamente” abierto desde afuera y a través del router, permanentemente monitoreado por *PortSentry* ya que según la configuración del ejemplo se encuentra monitoreando todos los puertos por debajo del

1024 (“Advanced Stealth Scan Detection Options”) y que no están siendo utilizados (se recuerda el cambio adrede del puerto de escucha del *ssh* a uno distinto del configurado por default). De esta forma actuará de manera activa (“Ignore Options”), generará una regla de dropeo de paquetes mediante *iptables* (“External Command”), agregará un registro en la tabla de ruteo descartando todo lo proveniente del host escaner (“Dropping Routes”) y por último insertará una línea al final del archivo */etc/hosts.deny* (“TCP Wrappers”) para marcar la IP del atacante como peligrosa y le deniegue el acceso. Todo esto (que en principio parece una configuración más que paranoica) se realiza en el mismo instante en que alguna máquina externa intente establecer una conexión a nuestro “ofrecido” puerto 22 (“Scan trigger value”)³⁷.

4.3.4 Resultados del ataque a Portsentry

Con este archivo de configuración y dos computadoras conectadas a internet se montó un laboratorio para comprobar que tan efectivo es este *anti-port-escanner* (ver Anexo II) pero lamentablemente las pruebas realizadas mostraron una gran falencia en la velocidad de reacción, retornando el resultado del barrido de puertos al atacante en cuestion de segundos, información que sería irrelevante para él con su IP bloqueada. Pero con técnicas conocidas (en este caso un *idle scan*³⁸ con *nmap [IDLESCAN]*) se pudo “engañar” al *Portsentry* para que bloquee la IP de una máquina comprometida. Concluyendo, ésta barrera solo frenaría a los atacantes más ingenuos, los cuales utilizan su propia IP para realizar dichos escaneos de puertos, por ende sirve (como se viene resaltando desde el primer capítulo de este trabajo de investigación) para establecer **una capa más** en nuestro sistema de seguridad y no la única barrera en la cual basar toda la seguridad de nuestro sistema.

4.4. Snort

Snort es un sistema de detección de intrusiones de red (*NIDS*) activo (que puede ejecutar acciones de respuesta), el cual implementa un motor de detección de ataques y barridos de puertos el cual permite registrar, alertar y reaccionar ante algún

³⁷ Hay que tener especialmente cuidado al intentar establecer una conexión desde fuera vía *ssh* ya que si se olvida especificar el puerto se bloqueará la IP y cualquier intento futuro de conexión será descartado.

³⁸ Para profundizar en el tema se recomienda la lectura del trabajo final de carrera de Marcelo Fernández, “Técnicas comunes de Ataque a equipos con Sistema Operativo Unix o derivados”, Capítulo 4.

intento de ataque.

Disponible bajo licencia GPL tanto para plataformas *Linux* y *Windows*, implementa el uso de una gran cantidad de patrones y filtros predefinidos al igual que *Nessus* para detectar posibles ataques o vulnerabilidades. Esta herramienta es muy común encontrarla implementada con componentes como **MySQL**³⁹ para almacenamiento de eventos y *ACID*⁴⁰ (*Analisis Control from Intrusion Database*) para ver y analizar luego los datos desde una interfaz web⁴¹.

Snort utiliza reglas escritas en un lenguaje sencillo, almacenadas en archivos de texto y agrupadas por categorías. Muchas de estas reglas son incluidas en la versión inicial de Snort y se pueden habilitar o ignorar desde el archivo *snort.conf*.

4.4.1 Estructura de las Reglas

Se basan en una serie de normas que se definen inicialmente como *Descripción*, *Cabecera* y *Opciones*. Luego de la descripción de la regla, se define en la cabecera la acción, protocolos, Ips, máscaras de subred, puertos origen y destino y los datos de destino del paquete. En la sección de opciones se contienen los mensajes y la información necesaria para tomar la decisión. Pero toda esta teoría se expone perfectamente en un ejemplo como puede ser que el IDS de aviso al detectar la descarga de un archivo *.mp3*. Una alternativa inicial y ciento por ciento didáctica es definir la siguiente regla:

```
Alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg "Se está descargando un
archivo mp3"; flags:AP; content:".mp3";)
```

Para comprender fácilmente este ejemplo es conveniente tener nociones de TCP/IP, concretamente los flags (ack, push, etc.). Para generar este tipo de reglas se puede utilizar un análisis de tráfico mediante Ethereal o TCPDump.

Otro ejemplo, en este caso que alerta un ataque de fingerprint mediante técnicas conocidas que utiliza *Nmap*⁴², es reaccionar ante el envío de los flags en conjunto de SYN – FIN – PUSH – URGENT.

39 Base de datos openSource (<http://www.mysql.org>)

40 Analysis Control for Intrusion Database (<http://www.cert.org/kb/acid>)

41 Junto con el servidor web Apache (<http://www.apache.org>)

42 NMAP utiliza esta técnica (<http://nmap.org/osdetect/osdetect-methods.html>) clasificada "T3 sends a TCP packet with the SYN, FIN, URG, and PSH flags set and a window field of 256 to an open port. The IP DF bit is not set." para detectar el Sistema Operativo remoto.

```
scan.rules:alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap fingerprint attempt";flags:SFP; reference:arachnids,05;)
```

4.4.2 Componentes de Snort

Snort se encuentra dividido en varios componentes que interactúan entre sí y forman un IDS en su totalidad.

- Decodificador de paquetes:
- Preprocesadores:
- Motor de detección:
- Sistema de alerta y logueo:
- Módulos de salida

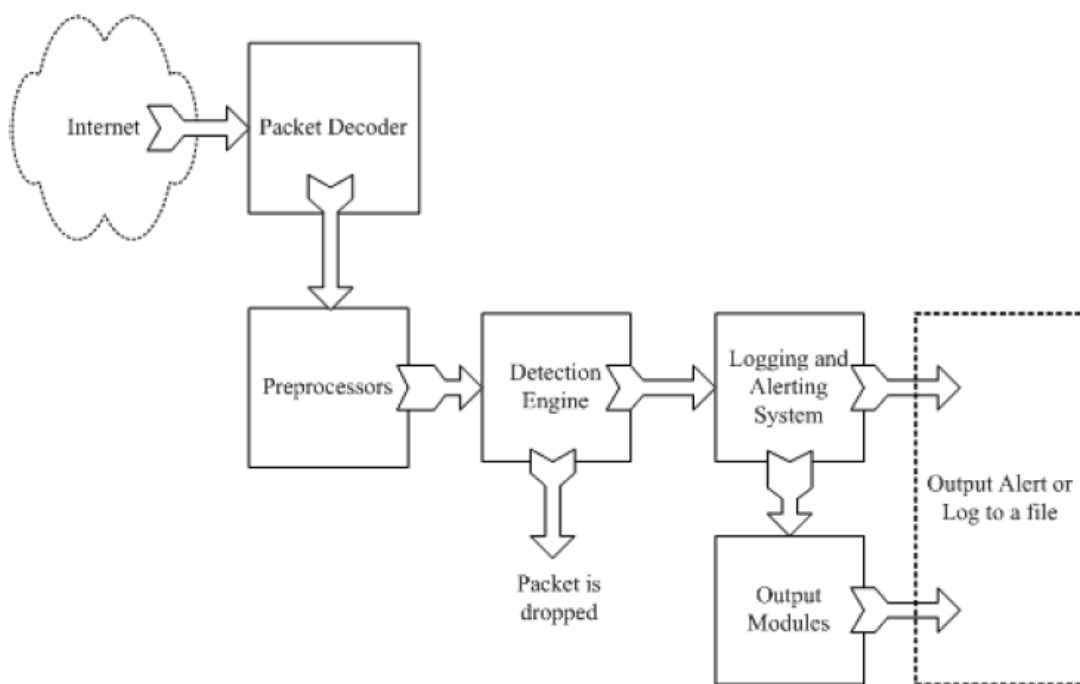


Figure 4-4 – Componentes de Snort

Una breve reseña de los mismos es presentada a continuación:

4.4.2.1 Decodificador de paquetes

Este decodificador toma los paquetes provenientes de las interfaces de red (no importando su tipo), los prepara y envía al preprocesador o al motor de detección.

4.4.2.2 Preprocesador

Los preprocesadores son módulos o *plugins* que se pueden utilizar con *Snort* para modificar los paquetes de datos antes de que sean atrapados por el motor de detección. Los preprocesadores pueden defragmentar paquetes, decodificar URLs HTTP, reensamblar streams TCP, etc.

Un ejemplo de preprocesador es el Arpspoof, el cual dispone de cuatro mecanismos de detección:

- Para cada una de las peticiones ARP detectadas se valida la dirección origen de Ethernet contra la dirección origen del paquete ARP.
- Para cada una de las respuestas ARP, se comparan las direcciones origen y destino, si alguna de estas no coincide *Snort* genera una alerta.
- También se generan alertas para peticiones ARP unicast (en lugar de broadcast).
- Se pueden comprobar los paquetes ARP basados en una lista de direcciones IP/MAC previamente generada por el administrador (se puede implementar solo en redes pequeñas).

Este es un ejemplo práctico de lo que puede hacer un preprocesador, aunque para el caso de este tipo de ataques la capacidad de *Snort* se denota que es limitada al igual que los sistemas mencionados en el capítulo anterior.

4.4.2.3 Motor de detección

Éste es el componente más importante de *Snort*, responsable de detectar si existe alguna actividad de intrusión en los paquetes. Este motor emplea las reglas descritas anteriormente y verifica si la condición de las mismas *matchean* con alguno de los paquetes observados. De allí se determina la acción a seguir que puede ser tanto un *logueo* en archivo o base de datos como el envío de una alerta al administrador. El tiempo de análisis de los paquetes es un factor crítico dentro del sistema, el cual depende directamente con la cantidad de reglas analizadas, la potencia de la máquina que se encuentra corriendo *Snort* y la carga de la red, entre otras cosas.

4.4.2.4 Sistema de alertas y logueo

Dependiendo de lo que el motor de detección encuentra en los paquetes, este sistema generará alertas o salidas al *log* del *Snort* (ubicado generalmente en */var/log/snort*⁴³) inicialmente configurado para mantener dos formatos: Un aviso en

43 Se puede modificar la ubicación de los archivos de logs mediante el parámetro *-l*.

texto claro y un detalle del tráfico en formato *tcpdump*.

4.4.2.5 Módulo de salida

También llamados *plugins de salida*, pueden realizar diferentes operaciones de envío de alertas o logueo de eventos, dependiendo básicamente de la configuración del sistema. Los más utilizados son:

- Logueo simple en archivo de texto (Ej. */var/log/snort/alert*)
- Intercambio de información a través de mensajes SNMP⁴⁴.
- Envío de mensajes al *syslog*.
- Logueo de eventos en bases de datos como MySQL u Oracle.
- Generación de XMLs.
- Modificación de reglas de ruteo o firewalls.
- Envío de mensajes a máquinas Windows a través de SMB.
- Y muchas otras herramientas como la utilización de alertas por mails o SMSs.

4.4.2.6 Resumen

En definitiva se puede concluir que estos cinco componentes en primer lugar preparan los paquetes a procesar^{4.4.2.1}; normaliza los headers, detecta anomalías, reensambla paquetes y streams de TCP^{4.4.2.2}; aplica las reglas a los paquetes^{4.4.2.3}; genera logs y alertas^{4.4.2.4}; y por último procesa estas salidas para generar un reporte final^{4.4.2.5}.

4.4.3 Protección del IDS

Una de las tareas más importantes es mantener protegido el sistema de detección de intrusos, ya que si el mismo se ve comprometido de alguna manera, todo el sistema lo estará, ya que el atacante podría empezar a generar falsas alarmas o incluso desactivar el Snort para realizar una intrusión de manera tranquila. Para evitar este tipo de inconvenientes se detallan a continuación algunas recomendaciones básicas:

- El IDS debe contar con un servidor de uso exclusivo donde no se ejecuten procesos adicionales ya que dichos servicios puede ser propensos a ataques.
- Se debe mantener actualizado el sistema continuamente.
- Configurar la máquina para que no responda pings.

44 SNMP o Protocolo Simple de Administración de Red es un protocolo de capa siete que facilita el intercambio de información de administración entre dispositivos de red.

- Implementarle alguna política de firewalls con Iptables/Netfilter en el caso de sistemas basados en Linux o Pf para el caso de openBsd.

A estas recomendaciones básicas se les puede agregar dos formas de ejecutar Snort de manera segura:

4.4.3.1 Snort en una interfaz Stealth

Se puede llegar a implementar un IDS basado en *Snort* conectándolo a una interfaz que solo tolere tráfico entrante mediante una interconexión a la red por un cable especial donde se encuentren cortados los terminales 1 y 2 mientras se mantienen los hilos de recepción en los terminales 3 y 4.

4.4.3.2 Snort en una interfaz sin IP asignada

También se puede implementar este IDS en una máquina con una interfaz de red sin IP asignada, de esta manera se protege del acceso de intrusos al Snort ya que el mismo no tiene una dirección IP.

4.5. Conclusión

Junto con este capítulo se completan todas las formas de monitoreo y técnicas de detección que son necesarias para afrontar las diferentes variantes de ataque e intrusión con métodos implementados, basados en el análisis de tráfico de red y con capacidades de reacción activa ante un evento anómalo.

5. Firewall

Un cortafuegos o *firewall* en inglés, es un sistema ubicado entre dos redes de computadoras para ejercer políticas de seguridad, controlando las comunicaciones entre los lados permitiendo o prohibiendo las mismas según lo establecido en las reglas definidas. Generalmente estos elementos se conectan en la intersección existente entre la red interna e internet, para que accesos no autorizados del exterior puedan aprovechar vulnerabilidades de la LAN. Otro método de conexión es unir tres redes creando una zona desmilitarizada o DMZ donde se ubican los servidores que deben ser accedidos desde el exterior como servicios web, ftp o correo (ver capítulo anterior XX).

5.1. Tipos de firewall

Por su arquitectura o forma de implementación [WWWSEGINF], los firewalls se pueden clasificar de las siguientes formas [WWWLEAR] :

5.1.1 Filtrado de paquetes, Screening Router o de capa de red:

Este tipo de firewalls posee uno de los modelos más sencillos y antiguos [SCHI97] en la capacidad de realizar un enrutado selectivo, trabajan a nivel de Transporte y de Red del modelo OSI y se encuentran conectados en ambos perímetros de la red. A este nivel de pueden realizar filtros según los campos de los paquetes IP (dirección IP de origen y dirección IP destino), a nivel cuatro (puerto origen y puerto destino) y a nivel de enlace de datos o capa dos (dirección MAC) [BUILDFW].

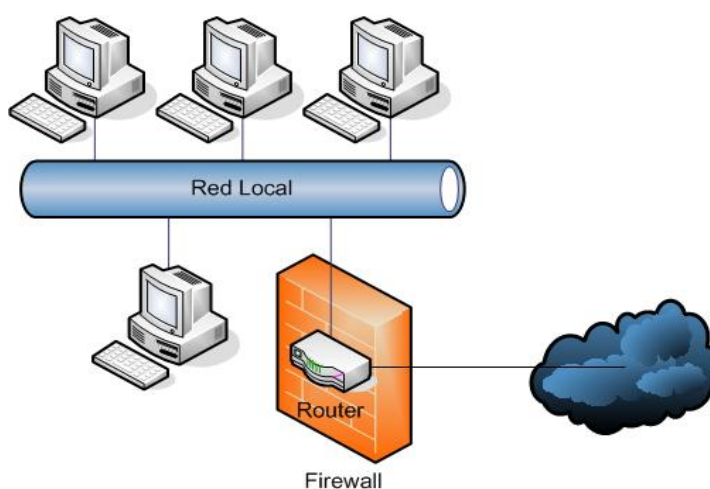


Figura 5.1 – Screening Router

Presenta limitaciones en cuanto a la protección de capas superiores a nivel OSI y

soporte de políticas de seguridad complejas como autenticación de usuarios o listas de control de acceso. Como contraparte la gran ventaja es que suelen ser económicos, transparentes para los usuarios y poseen un alto nivel de desempeño [WWTEXTCI]

5.1.2 Proxy-Gateway o de capa aplicacion:

Este tipo de firewall trabaja a nivel de aplicación o capa 7 del modelo OSI de manera que puede implementar filtrados específicos aprovechando características del protocolo de este nivel. La utilización de este servidor intermediario entre el cliente y el servidor real sigue siendo transparente para el usuario. Como ejemplo, un servidor de proxy cache permite a un conjunto de computadoras establecer una conexión web limitando el tráfico HTTP mediante su análisis, restringiendo los accesos según URLs o contenido que violen las reglas de seguridad. Una buena implementación de un proxy cache es la utilización de la herramienta open source Squid, disponible en <http://www.squid-cache.org> en su versión 3.0 (estable desde diciembre de 2007), o desde los repositorios oficiales de Ubuntu en su versión 2.6-14. Squid permite añadir seguridad a nivel de filtrando de tráfico, estableciendo reglas de control de acceso a partir de los archivos de configuración (generalmente ubicados en */etc/squid/squid.conf*). Squid se puede combinar con la aplicación *SquidGuard*⁴⁵ para facilitar el manejo de prohibición de acceso a páginas mediante el uso de listas negras⁴⁶ simplemente modificando una línea en el archivo de configuración de Squid.

```
redirect_program /usr/bin/squidGuard -c /etc/squid/squidGuard.conf
```

Una vez obtenida la lista negra, se deberá copiar en el directorio especificado en el archivo de configuración de *squidguard* (generalmente ubicado en */etc/squid/squidGuard.conf*) que por defecto posee las siguientes líneas definidas:

```
dbhome /var/lib/squidguard/db
logdir /var/log/squid
```

45 Disponible en <http://www.squidguard.org>

46 BlackLists:

- MESD [<http://squidguard.mesd.k12.or.us/blacklists.tgz>]
- Shalla's [<http://www.shallalist.de>]
- Université Toulouse [http://cri.univ-tlse1.fr/documentations/cache/squidguard_en.html#contrib]
- Blacklist.com [<http://urlblacklist.com/?sec=download>]

Donde se definen en primer lugar la ubicación de las listas de páginas prohibidas y en segundo lugar donde serán escritos los mensajes del Squid.

Otro ejemplo no tan divulgado como el anterior pero no por eso menos importante es la implementación de un servidor de mail relay para actuar en forma similar al proxy web pero con el tráfico SMTP para de esta manera eliminar los mails denominados SPAM entrantes mediante diversas reglas antes de que sean entregados a los mailbox finales⁴⁷ [BUILDFW].

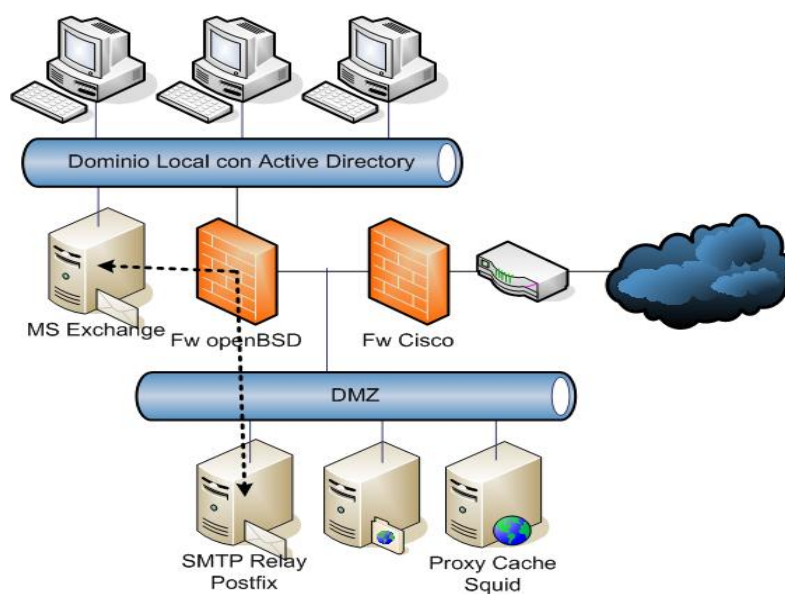


Figura 5.2 - Ejemplo de una arquitectura heterogénea con SMTP relay.

5.1.3 Dual Homed Host

Esta arquitectura consiste en un único equipo que implementa funciones de filtrado tanto a nivel de red como de aplicación, mediante el uso de dos interfaces [SIHA95] conectadas lógicamente y físicamente a ambos segmentos de red distintos (la red interna o privada y la red externa o internet). Al igual que la plataforma anterior, la clave de seguridad se encuentra en bloquear el ruteo de paquetes de forma directa obligando establecer las conexiones siempre usando este *proxy* o *host bastion* como intermediario (es por eso que el sistema debe ejecutar por lo menos un servidor proxy por servicio). Esta arquitectura es mucho más segura que la anterior aunque igualmente

⁴⁷ Si por cuestiones meramente comerciales o de compatibilidad con productos impuestos a nivel comercial, una organización debería implementar servicios no recomendables, es una buena práctica colocar este tipo de servidores “relay” en la DMZ que funcionan a su máxima capacidad, aliviando la carga de trabajo de los siguientes servidores y sobre todo aislando los productos vulnerables u obsoletos desde el exterior.

presenta grandes desventajas ya que si el servidor es traspasado la totalidad de la red interna queda sin protección [BUILDFW].

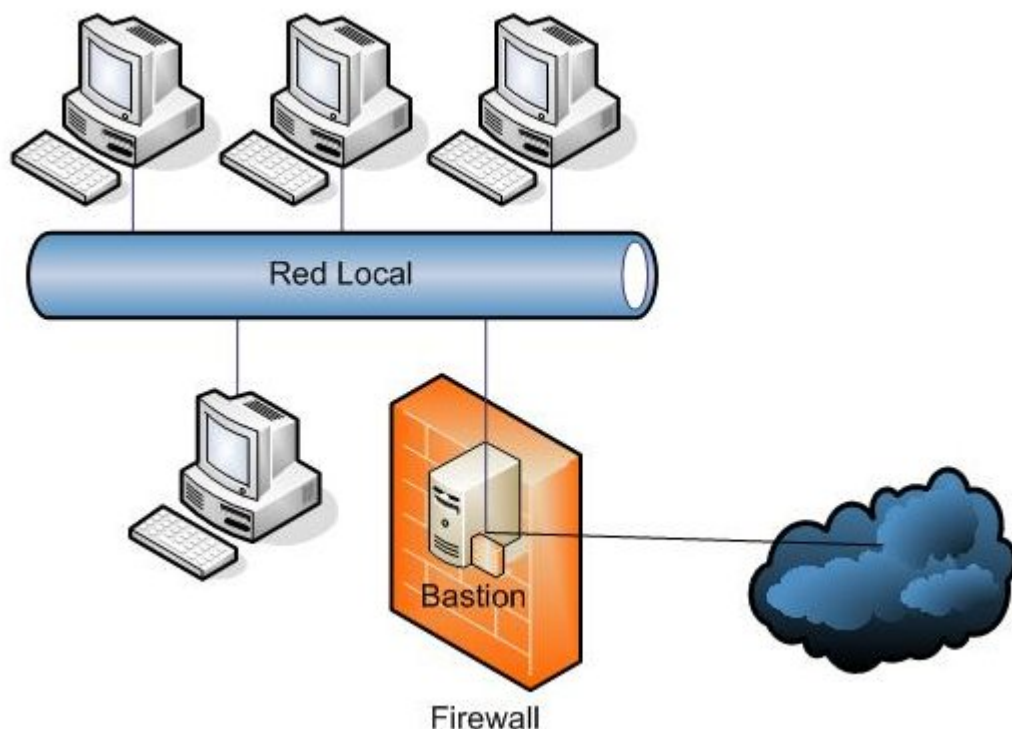


Figura 5.3 – Dual Homed Host.

5.1.4 Screened Host

Para escalar un nivel más en términos de seguridad, se presenta la arquitectura de screened host, donde son combinadas dos de las alternativas anteriores (Screening Router y Dual Homed Host) donde en una primera instancia se filtran paquetes mediante el *router* y en la segunda línea de defensa se sitúa el *host bastion* con un reducido número de servicios publicados hacia el exterior. Entonces, el router se encuentra configurado para dirigir todo el tráfico de la red externa al host bastion (lo único que se puede acceder desde el exterior, y este mismo dirige todo el tráfico de la red interna hacia el router (único vínculo con el exterior) [BUILDFW].

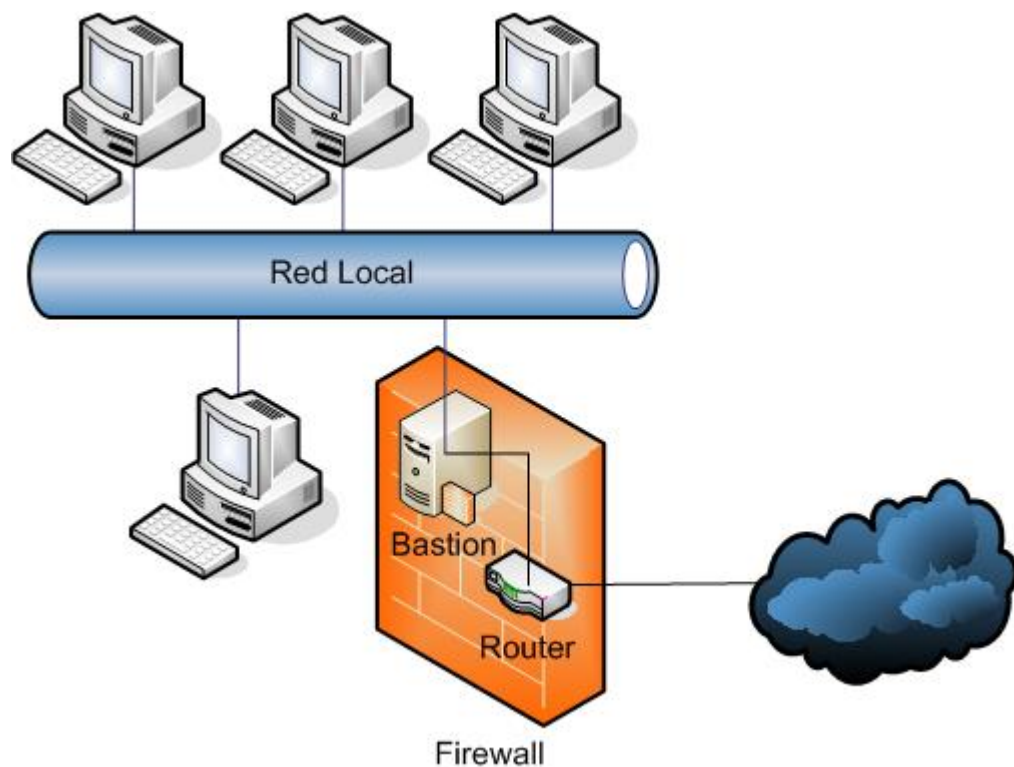


Figura 5.4 – Screened host

Ahora, existen dos alternativas para situar el router: Antes o después del host bastion. Muchos autores [MRAN93] [CSEM96] recomiendan ubicar el router como se indica en la figura anterior (entre la red externa y el *host bastion*, pero otros [WACA94] defienden exactamente la idea contraria sosteniendo que el router en medio no provoca una degradación en la seguridad del sistema. De todas maneras es más fácil para el administrador del sistema instalar y administrar un router del lado externo que mantener un host en esa primera línea (por más que se intente, es muy difícil ofrecer menos servicios en un host "minimamente configurado" que en un IOS de algún router Cisco).

5.1.5 Screened Subnet

La arquitectura de *Screened Subnet*, o más conocida como *De-militarized Zone*, zona desmilitarizada o simplemente *DMZ*, es actualmente la más utilizada e implementada ya que incrementa el nivel de seguridad agregando una subred intermedia entre las redes externa e interna, donde se ubican los servidores y servicios públicos (como *http*, *ftp*, correo, etc.) y se evita tener un único punto débil como en el caso del

host bastion. Para este caso es necesario aumentar la complejidad de la configuración anterior ubicando dos router (uno en cada lado de los servidores públicos). El router conectado al exterior (generalmente internet) es el encargado de bloquear el tráfico no deseado entre la red perimetral y la DMZ, mientras que el router interior tiene el mismo objetivo pero entre la DMZ y la red interna. De esta forma, el atacante debe romper la seguridad de ambos routers para acceder a la red privada. También se puede aumentar los niveles de seguridad (a un modo paranoico) donde se definen varias redes perimétricas en serie desde los servicios que requieren menos fiabilidad y hacia la red interna [BUILDFW].

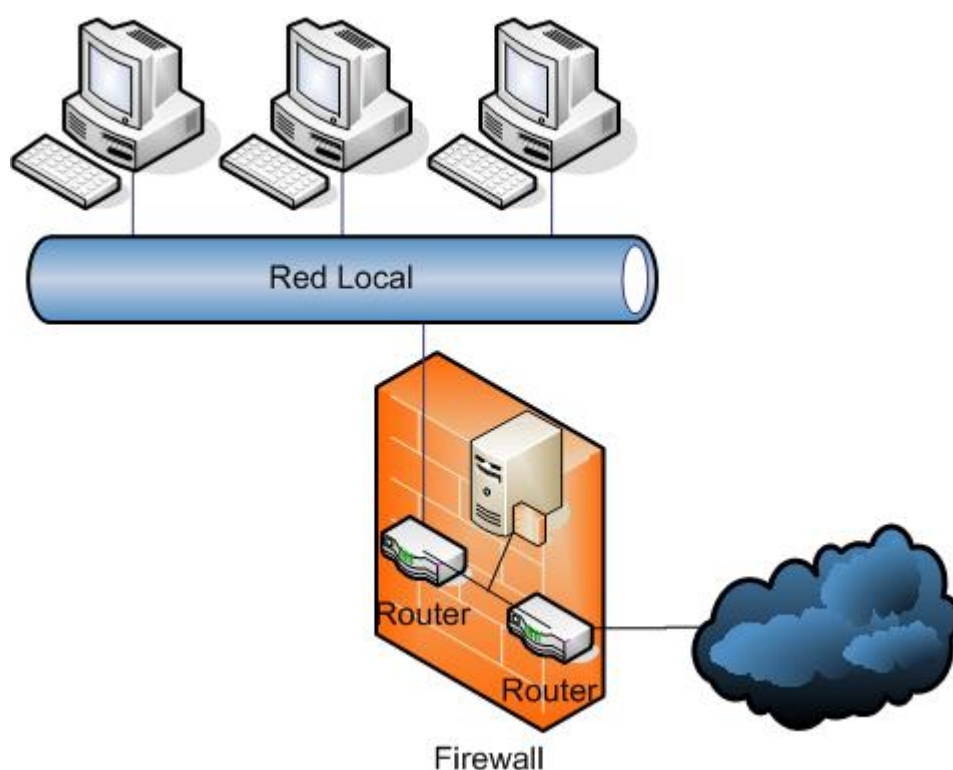


Figura 5.5 – Screened Subnet o DMZ

5.1.6 Firewall Personal

Un caso particular de firewall es el que se instala como software (diseñado para usuarios finales) en una máquina, donde se filtran las comunicaciones entre dicha computadora y el resto de la red. Varias aplicaciones son conocidas y probadas, generalmente interfaces gráficas que utilizan el sistema de *Iptables-Netfilter* incluido en el kernel Linux. Sin entrar en detalle de su instalación y uso⁴⁸, el Firestarter es una

48 Se puede extender el tema en <http://www.kriptopolis.org/firestarter>

herramienta de firewall personal y de código abierto que usa este tipo de sistemas y posee una aplicación gráfica de fácil configuración para definir reglas de bloqueo y otras opciones, además de permitir el monitoreo de todo el tráfico de red del sistema, eventos generados por ejemplo, por intentos de intrusión, redireccionamiento de puertos, compartición de conexiones de internet y el servicio de DHCP [BUILDFW].

5.2. Estrategias en diseños de firewalls

A la hora de configurar un firewall se debe tener en claro el paradigma o estrategia a seguir. Básicamente se dividen en dos:

- Se permiten todos los servicios, excepto los que están expresamente prohibidos.
- Se bloquean todos los servicios, excepto los que estén expresamente permitidos.

Esta última opción es la que recomiendan la mayoría de los autores, también conocida como “Denegación Predeterminada”, y como se mencionó anteriormente, deniega el paso de todo el tráfico y a partir de ahí va permitiendo el paso de los paquetes a través del firewall. Esta aproximación es la recomendada ya que cualquier descuido en la configuración del firewall estará cubierta por la restricción general [BUILDFW].

5.3. Netfilter/Iptables

Netfilter es un proyecto que presenta una infraestructura para el manejo de paquetes de red ofrecida en el núcleo de *Linux*. Dentro de la misma se encuentra *Iptables*, un componente de *Netfilter* que permite filtrar paquetes, realizar *Nat* de direcciones IP y registrar eventos en archivos logs.

5.3.1 Historia

En el año 1998, Rusty Russell inició el proyecto *Netfilter* y un año después fué incorporado en el *kernel 2.3* de Linux para comenzar a reemplazar los programas utilizados hasta entonces para crear firewalls (*ipchains* e *ipfwadm*). *Iptables* es una aplicación a nivel de usuario que permite configurar tablas, cadenas y reglas de *netfilter*.

5.3.2 Operación

Iptables define reglas que son agrupadas en cadenas, cada una de estas cadenas está formada por una lista ordenada de reglas y agrupada en tablas asociadas

con un tipo diferente de procesamiento de paquetes.

5.3.2.1 Tablas y cadenas

- **FILTER TABLE:** Responsable del filtrado de paquetes, los cuales si o si pasarán por alguna de las cadenas predefinidas.
- **INPUT:** El cual es atravesado por los paquetes que tienen como destino el propio sistema.
- **OUTPUT:** El cual concuerda con los paquetes que son generados por este sistema.
- **FORWARD:** El cual es recorrido en el caso de que los paquetes pasen por el sistema para ser ruteados al verdadero destino.
- **NAT TABLE:** Encargada de configurar las reglas de reescritura de direcciones o puertos. Las cadenas predefinidas se enumeran a continuación:
- **PREROUTING:** Atravesada por los paquetes entrantes antes de que se consulte la tabla de ruteo local (principalmente destination-NAT).
- **POSTROUTING:** Por donde circulan luego de haber elegido el camino de ruteo (destination-NAT).
- **OUTPUT:** Para los paquetes generados localmente, permitiendo hacer un destination-NAT.
- **MANGLE TABLE:** Esta tabla está diseñada para manejos avanzados como por ejemplo la calidad de servicio. Ya que todos los paquetes pasan por la mangle table todas las cadenas predefinidas están disponibles (*PREROUTING*, *INPUT*, *FORWARD*, *OUTPUT*, *POSTROUTING*).

5.3.2.2 Destino de las reglas

Cada una de las reglas puede tener diferentes destinos ya incorporados:

- **ACCEPT:** El cual, como su nombre lo indica, es aceptado por netfilter, donde dependiendo del caso de la cadena, puede que se apruebe su ingreso al host (*INPUT*), que se apruebe su egreso (*OUTPUT*) o bien sea aceptado su ruteo a través del host (*FORWARD*).
- **DROP:** Este destino indica que los paquetes deben ser descartados sin ningún tipo de procesamiento.
- **QUEUE:** Lo que hace es encolar los paquetes en espacio de usuario para que una aplicación personalizada los tome y modifique. Si no existe dicha aplicación, el tratamiento es similar a un *DROP*.
- **REJECT:** De similares características que *DROP*, con excepción de que envía al origen un paquete de error (como puede ser un *tcp-reset*, *icmp-protocol-unreachable* o un *icmp-port-unreachable*).
- **LOG:** Este destino se usa para enviar mensajes a un archivo y es generalmente utilizado para *debuggear* tráfico.
- **DNAT/SNAT:** Hacen que la dirección IP y opcionalmente el puerto sean

reescritos para la traducción de direcciones de red.

- **MASQUERADE:** Trabaja de manera similar a *SNAT* pero automatiza la traducción de la IP de salida (generalmente utilizado con direcciones públicas dinámicas).

5.3.2.3 Estado y seguimiento de las conexiones

El seguimiento de los paquetes es importante para llevar cuenta de las sesiones lógicas de la red y de este modo relacionarlos de manera que se puedan analizar las conexiones en su conjunto. Los cuatro estados en que se clasifican son:

- **NEW:** Creando una nueva conexión.
- **ESTABLISHED:** Forma parte de una conexión existente.
- **RELATED:** Relacionada pero no “parte de” una conexión.
- **INVALID:** No es parte de ninguna conexión existente y capaz de crear por sí misma una nueva conexión.

5.3.2.4 Parámetros más usados de iptables

- **-F** : Elimina todas las cadenas.
- **-X** : Para eliminar una cadena.
- **-Z** : Pone en cero el contador de paquetes.
- **-t** : El comando se aplica a una tabla especificada.
- **-P** : Especifica una política a una cadena específica.
- **-A** : Agrega una regla a una cadena
- **-i** : Especifica una interfaz a través de la cual un paquete será recibido.
- **-o** : Especifica una interfaz a través de la cual un paquete será enviado.
- **-j** : Especifica el destino de una regla.
- **-D** : Elimina una regla de una cadena.
- **-s** : Especifica una dirección IP de origen con la que debe matchear la regla.
- **-d** : Especifica una dirección IP destino con la que debe matchear la regla.
- **-m** : extiende la regla de matcheo.
- **-p** : Especifica un protocolo (icmp, udp, tcp)
- **--dport** : Matchea paquetes que contienen como destino los puertos indicados.
- **--sport** : Matchea paquetes que contienen como origen los puertos indicados.
- **--state** : Especifica el estado que deben tener las conexiones (*RELATED*, *ESTABLISHED*)
- **-L** : Lista las reglas actualmente implementadas.

Dos comandos importantes son iptables-save para generar una salida con las reglas implementadas en la sesión actual e iptables-restore para lograr el caso inverso.

5.3.3 Implementación genérica con DROP por defecto

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F
## Se Establece politica por defecto: DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
## Debemos decir de manera explicita qué es lo que queremos abrir
# Se acepta todo (entrante y saliente) del localhost
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
# Lo mismo (pero de diferente modo) para nuestra IP
iptables -A INPUT -s 10.218.106.100 -j ACCEPT
iptables -A OUTPUT -d 10.218.106.100 -j ACCEPT
# Se permite que la maquina pueda salir a internet
iptables -A INPUT -p tcp -m tcp --sport 80 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
# Y tambien a webs seguras
iptables -A INPUT -p tcp -m tcp --sport 443 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 443 -j ACCEPT
# Reglas necesarias para FTP pasivo y activo.
iptables -A INPUT -p tcp -m tcp --sport 20:21 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 20:21 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --sport 1024:65535 --dport 1024:65535 -m state --state
ESTABLISHED -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --dport 1024:65535 -m state --state
NEW,RELATED,ESTABLISHED -j ACCEPT
# Se permite la consulta a los DNS
iptables -A INPUT -s 10.218.106.31 -p udp -m udp --sport 53 -j ACCEPT
iptables -A OUTPUT -d 10.218.106.31 -p udp -m udp --dport 53 -j ACCEPT
iptables -A INPUT -s 10.218.106.32 -p udp -m udp --sport 53 -j ACCEPT
iptables -A OUTPUT -d 10.218.106.32 -p udp -m udp --dport 53 -j ACCEPT
# Se permite la consulta al servidor de tiempo
iptables -A INPUT -s 10.218.106.1 -p udp -m udp --dport 123 -j ACCEPT
iptables -A OUTPUT -d 10.218.106.1 -p udp -m udp --dport 123 -j ACCEPT
# Barrera de backup por si cambiamos a modo ACCEPT temporalmente
# Con esto protegemos los puertos reservados y otros well-known
iptables -A INPUT -p tcp -m tcp --dport 1:1024 -j DROP
iptables -A INPUT -p udp -m udp --dport 1:1024 -j DROP
iptables -A INPUT -p tcp -m tcp --dport 1723 -j DROP
```

```
iptables -A INPUT -p tcp -m tcp --dport 3306 -j DROP
iptables -A INPUT -p tcp -m tcp --dport 5432 -j DROP
# Fin del script
```

5.4. openBSD - Packet Filter (*pf*)

5.4.1 Plataforma

Antes que nada, para construir un sistema seguro bueno sería que el firewall perimetral esté montado sobre una plataforma indestructible [WIKIOPENBSD]. Como se comentó reiteradas veces, el sistema indestructible no existe, pero buenas son las estadísticas que ostenta *openBSD* [OPENBSD.ORG] [WIKIOBSD] para montar un firewall estable sobre este sistema operativo que publicó en su sitio oficial las siguientes leyendas:

- Hasta el 2002: «Ningún fallo de seguridad remoto en la instalación por defecto en los últimos 6 años»⁴⁹.
- hasta el 2004 : «Un sólo agujero de seguridad en la instalación por defecto, en más de 8 años»⁵⁰.
- hasta la actualidad : «Solo dos agujeros de seguridad en la instalación por defecto, en más de 10 años»

El *openBSD*⁵¹ es un sistema operativo libre⁵² originado por *Theo de Raadt*, del tipo Unix, basado en el 4.4BSD y descendiente directo del netBSD⁵³ enfocándose desde sus raíces en la seguridad del sistema.

5.4.2 Objetivos del Proyecto OpenBSD

Los objetivos de mayor relevancia se encuentran en la página del proyecto,

49 A partir de un agujero de seguridad encontrado en el OpenSSH.

50 Después de que se encontrara un fallo en el módulo de Ipv6.

51 [Http://www.openbsd.org](http://www.openbsd.org)

52 Según Theo, *"el software que OpenBSD utiliza y distribuye debe ser libre para todos (tanto usuarios como compañías), para cualquier fin que quiera dársele, incluyendo su modificación, uso, mear sobre él o incluso su integración en máquinas trituradoras de bebés o en bombas atómicas para arrojar en Australia"*

53 de Raadt generó un fork de netBSD a partir de diferencias filosóficas y personales con los demás miembros de netBSD.

pero a nivel global se pueden destacar los siguientes [SEGPBOSD] [WWWBOSD]:

- Disposición total de acceso a los fuentes del proyecto.
- Integración de cualquier tipo de código que disponga de una licencia aceptable para que, como se dijo anteriormente, el usuario *haga lo que quiera o necesite* con ese código.
- Llevar un seguimiento total de posibles bugs y fallos, y corregirlos antes que nadie para conseguir un sistema operativo lo más seguro posible.
- Seguir siempre e implementar en la medida de lo que sea posible los estándares⁵⁴.
- Ser un proyecto totalmente apolítico.

El enfoque en este sentido, claro desde las primeras versiones, ha mejorado *release* a *release* e incluyó grandes ideas como la generación aleatoria de PIDs⁵⁵ (Process ID o ID de proceso), generación aleatoria de los números de secuencia iniciales de las conexiones TCP, integración de algoritmos de criptografía en el kernel, separación de los privilegios para los demonios del sistema, uso de *systrace* (monitor de acceso a llamadas del sistema) o *propolice* (protección de la pila), etc. [BSDSCRU]

Además de buscar fallos y auditar el código constantemente a medida que se encuentran nuevos fallos y bugs, y explorando con técnicas que se van descubriendo día a día en el campo de la seguridad informática, lo que logra una metodología proactiva en constante fase de auditoría, aumentando la posibilidad de ser los primeros en encontrar un fallo en cualquier parte del sistema.

5.4.3 Packet Filter

Packet Filter (a partir de ahora *pf*) es el filtro de paquetes de *openBSD*, escrito originalmente por *Daniel Hartmeier* como reemplazo del anterior módulo desarrollado por *Darren Reed*, el *IPFilter*⁵⁶ [COD23].

Algunos de los puntos fuertes de *pf* son [OBSD_PF]:

- Filtrado de paquetes Ipv4 e Ipv6.

54 Como ANSI, POSIX, etc.

55 Número con el que se puede referenciar a cada uno de los procesos del sistema.

56 El cambio de esta aplicación se debió a un problema en la licencia de IPFilter, ya que Darren debía dar autorización a los desarrolladores de openBSD por cada cambio en el código.

- NAT (Network Address Translation).
- Normalización de paquetes.
- Set de reglas dinámicos.
- Balanceo de carga.
- Modulación de ancho de banda.
- Logueo de paquetes.
- Autenticación de usuarios contra el firewall.

5.4.4 Sincronización de reglas y redundancia:

Para mantener la disponibilidad, es fundamental que el firewall no sea un único punto de fallo. Desde la versión 3.5 de openBSD se han incorporado varios componentes para resolver este inconveniente. La solución es implementar dos (o más) firewalls en paralelo: uno primario y otro de resguardo. Cuando el primario cae, el segundo firewall toma su lugar asumiendo la identidad del “de producción” y continuando su trabajo donde éste lo dejó, preservando las conexiones existentes y continuando con el tráfico de la red como si nada hubiera pasado [SCIRET] [OBSD_PF].

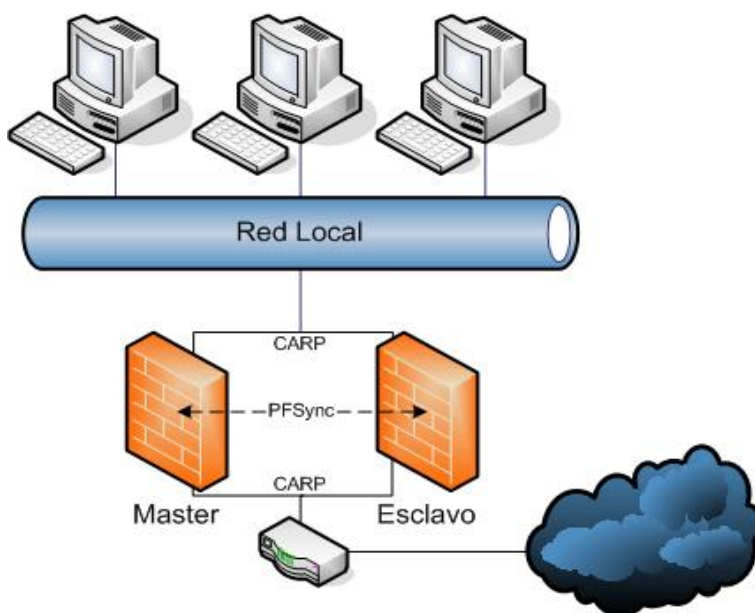


Figura 5.6 – Ejemplo de configuración de firewall redundante.

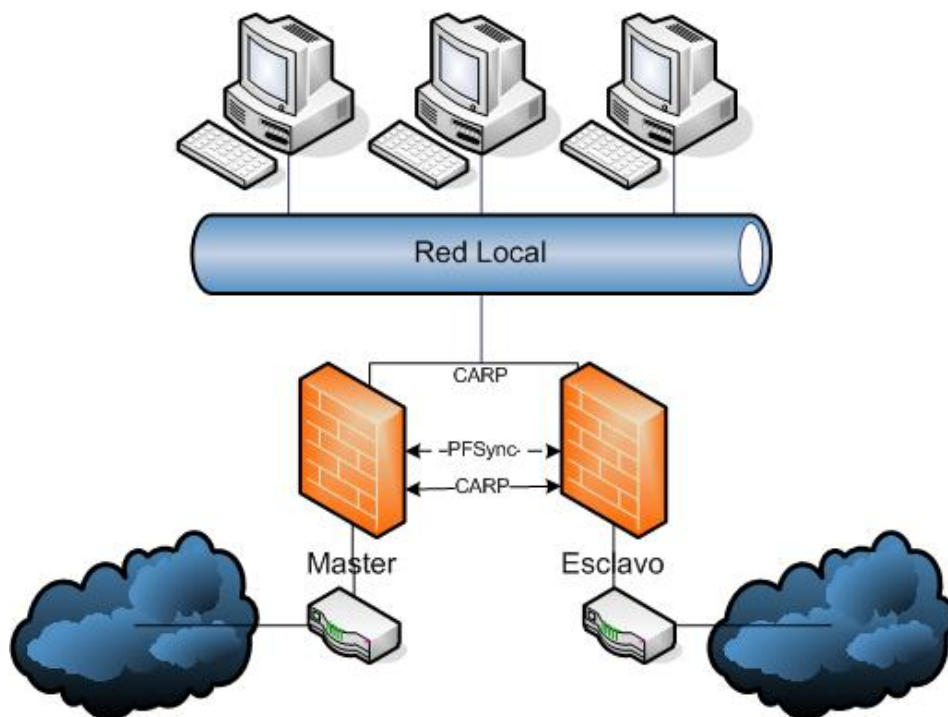


Figura 5.7 – Ejemplo de configuración de firewall redundante.

Para realizar esta acción openBSD utiliza dos componentes básicos: *CARP* (o Common Address Redundancy Protocol) [CARP], que permite a una máquina back-up (controlando los fallos en las capas 2 y 3 del modelo OSI) asumir la identidad del primario⁵⁷; y *PFSYNC*, que asegura sincronizar el estado del firewall para poder retomar las conexiones donde el anterior lo dejó⁵⁸. Para garantizar la seguridad de este micro-sistema, es recomendable utilizar un cable cruzado creando una red dedicada entre dos interfaces extras.

5.4.5 Proxy TCP Syn

Generalmente cuando un host inicia una conexión TCP con un servidor externo, *PF* reenvía la comunicación inicial (o handshake) entre ambos extremos. Pero también puede ser configurado para que haga de *proxy* separando las partes, completando el saludo inicial con cliente, iniciando un saludo inicial con el servidor externo, y pasando paquetes entre ambas partes eliminando la amenaza de

⁵⁷ El servidor primario manda mensajes de aviso vía multicast usando un protocolo CARP de forma regular, el host backup escucha estos avisos y si los mismos se detienen el secundario se da cuenta para poder tomar el control de las conexiones.

⁵⁸ pfsync transfiere mensajes de estado de inserciones, actualizaciones y borrados entre los firewalls vía multicast utilizando el protocolo PFSYNC.

desbordamientos de TCP SYN falseados (técnica por demás conocida para hacer caer un servidor) [OBSD_PF].

Para activar esta opción solo se debe agregar una línea en las reglas de filtrado con la opción *synproxy state*.

```
pass in on $ext_if proto tcp from any to $web_server port www \ flags S/SA  
synproxy state
```

5.4.6 Priorizando ACK con ALTQ (alternate Queueing)

El regulador de caudal de tráfico ALTQ también está incluido en el pf de openBSD y está formado por un conjunto de herramientas que permiten manejar QoS (calidad de servicio) armando colas de tráfico, manejando asignación de caudales y modificando prioridades de paquetes. Desde la *versión 3.3 de openBSD* está integrado este módulo que permite modelar las colas de paquetes que son almacenados en el firewall de forma que, en lugar de seguir el orden FIFO (el primero que llega es el primero en salir) se puedan establecer reglas y prioridades basados en el origen, destino o tipos de paquetes que se están tratando [PRIORACK] [ACKPRIOR] [OBSD_PF].

Un problema al que se le puede dar fin utilizando ALTQ es por ejemplo, cuando se degrada el servicio si se mantiene una conexión X (donde continuamente se intercambian ACKs con el otro extremo) y al mismo tiempo se hace un *upload* de un archivo utilizando una conexión Y. El sistema de origen tendría que responder las peticiones del cliente lo más rápidamente posible, y por ende, los ACKs de la descarga entrarán en una cola de espera haciendo creer al sistema remoto que el propio es muy lento para responder, lo que generará una negociación del tamaño de la ventana de transmisión y empezará a relentizar la conexión de descarga.

La solución que puede utilizar pf mediante ALTQ es priorizar los ACKs, dándole mayor importancia al proceso de respuesta en la comunicación.

Para lograr esto, ALTQ utiliza tres tipos de algoritmos:

- **cbq (class based queueing)** : Arma colas basadas en clases, organizadas en una estructura de árbol (puede tener subcolas), cada una de las cuales puede tener definido un ancho de banda y una prioridad.
- **priq (priority queueing)**: Este algoritmo no permite armar subcolas, simplemente le asigna a las colas una prioridad de 0 a 15 donde se tratan las prioridades más altas primero.

- **hfsc (Hierarchical Fair Service Curve):** Similar a cbq salvo que además aporta compartición de enlaces y garantiza el servicio en tiempo real gracias a la utilización de una curva de servicio basada en el modelo QoS.

Ahora si, se puede armar un ejemplo con reglas sencillas y de fácil lectura como es de costumbre para *packet filter*.

```
#Se define la interfaz externa
ext_if="pcn0"
#Se habilita ALTQ en dicha interfaz
#con el scheduler PRIQ y definiendo 230Kbps de subida.
altq on $ext_if priq bandwidth 230Kb queue { q_pri, q_def }
#Se define una cola con prioridad 7
queue q_pri priority 7
#Se define otra cola con prioridad 1 (default)
queue q_def priority 1 priq(default)
#Se especifican los paquetes que pueden salir siempre}
#Si son tcp y llevan la bandera S/SA. asignele las colas definidas.
pass out on $ext_if proto tcp from $ext_if to any flags S/SA \
keep state queue (q_def, q_pri)
#Lo mismo pero de entrada
pass in on $ext_if proto tcp from any to $ext_if flags S/SA \
keep state queue (q_def, q_pri)
```

5.4.7 Normalización de paquetes irregulares

La normalización de paquetes irregulares o *scrubbing* tiene por objetivo establecer que no existan ambigüedades con la interpretación del paquete en su destino final. Esta directiva, en su uso más explotado, realiza el reensamblaje de paquetes fragmentados para proteger ciertos tipos de ataque bloqueando los paquetes *tcp* que posean un indicador no válido en su estructura [BSDSCRU] [OBSD_PF].

Para normalizar la entrada de todos los paquetes en todas las interfazs se debe agregar la siguiente línea en el *pf.conf*:

```
scrub in all
```

Además, a continuación se detallan las opciones más usadas que pueden acompañar esta directiva:

- **no-df** : Elimina el bit de don'tfragment de la cabecera IP.
- **random-id** : Reemplaza el campo de identificación IP con valores aleatorios.
- **min-ttl num** : Impone un *tll* (*time to live*) mínimo en las cabeceras IP.

- **max-mms num** : Impone un *mms* (*maximun segment size*) máximo en las cabeceras IP.
- **fragment reassemble** : Almacena en memoria fragmentos de paquetes entrantes y los reensambla armando un paquete completo antes de pasarlo.
- **fragment crop** : Impone que se rechacen fragmentos duplicados.
- **reassemble tcp** : Normaliza las conexiones *tcp* manteniendo su estado.

Muchas de estas opciones se aplicaron en pruebas de laboratorio explicadas en el Anexo IV.

5.4.7.1 Ejemplos de utilización de scrub:

```
scrub in on fxp0 all fragment reassemble min-ttl 15 max-mss 1400
scrub in on fxp0 all no-df
scrub on fxp0 all reassemble tcp
```

5.4.8 Implementación de un firewall básico con pf

A continuación se detalla la implementación de un *firewall básico*⁵⁹ con reglas de *pf* que ya fueron explicadas anteriormente. Hay que resaltar que a diferencia *iptables*, la última regla que *matchea* con un paquete es la vencedora [SAMPPF] [FWOBSD] [OBSD_PF].

```
# Configuracion
ext_if="fxp0"
int_if="xl0"
tcp_services="{ 22, 113 }"
icmp_types="echoreq"
comp3="192.168.0.3"

# Opciones
set block-policy return
set loginterfaz $ext_if
set skip on lo

# scrub (para normalizar la entrada de paquetes)
scrub in

# nat/rdr
nat on $ext_if from !($ext_if) -> ($ext_if:0)
```

⁵⁹ Ver Implementación paranoica de un firewall con packet filter en:
http://wiki.sblug.org/Example_pf.conf_for_OpenBSD_Firewall

```
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
rdr pass on $int_if proto tcp to port ftp -> 127.0.0.1 port 8021
rdr on $ext_if proto tcp from any to any port 80 -> $comp3

# Reglas de filtrado
block in
pass out keep state
anchor "ftp-proxy/*"
antispoof quick for { lo $int_if }
pass in on $ext_if inet proto tcp from any to ($ext_if) \
    port $tcp_services flags S/SA keep state
pass in on $ext_if inet proto tcp from any to $comp3 port 80 \
    flags S/SA synproxy state
pass in inet proto icmp all icmp-type $icmp_types keep state
pass in quick on $int_if
```

5.4.9 Resumen de uso del firewall

Para obtener el detalle de la información sobre el uso de *pf* en el sistema, es muy útil tener en cuenta la siguiente expresión que se puede ejecutar desde la línea de comandos del openBSD:

```
$ sudo pfctl -s info
Status: Enabled for 0 days 00:06:12          Debug: Urgent

Hostid: 0x71019b0e

State Table                                Total          Rate
current entries                            0
searches                                  174             0.5/s
inserts                                    0             0.0/s
removals                                    0             0.0/s
Counters
match                                      174             0.5/s
bad-offset                                  0             0.0/s
fragment                                    0             0.0/s
short                                       0             0.0/s
normalize                                    0             0.0/s
memory                                      0             0.0/s
bad-timestamp                              0             0.0/s
congestion                                  0             0.0/s
ip-option                                    4             0.0/s
proto-cksum                                 0             0.0/s
state-mismatch                              0             0.0/s
state-insert                                0             0.0/s
```

state-limit	0	0.0/s
src-limit	0	0.0/s
synproxy	0	0.0/s

5.5. Conclusión

Después de instalar y aplicar ambos métodos de filtrados de paquete, quedó a la vista la simplicidad y potencia del segundo sistema de *firewall*, basado en *openBSD* y *packet filter*, por encima de la combinación *Linux/Netfilter-Iptables*. Aunque esta última opción no es para nada descartable y cambia radicalmente su usabilidad si se combina con aplicaciones que permiten obtener una abstracción en la escritura de reglas de filtrado (como firewalls ya prearmados configurables o aplicaciones de usuario como firestarter).

6. Seguridad Wi-Fi

En este capítulo se analizará el nivel de seguridad que brindan las *redes wireless*, la realidad actual, los métodos de encriptación de que se dispone, y hacia donde están evolucionando los mismos. Con solo tomar conciencia que en una red inalámbrica se tiene resuelto el problema de la necesidad de tener una conexión física para capturar el tráfico en la red, y considerando que cualquier persona ubicada a menos de cien metros se encuentra al alcance de la misma, es erróneo no pensar en utilizar algún método de encriptación para una conexión de este tipo. Sin embargo esto no es considerado por la mayoría de los usuarios de esta nueva arquitectura que instalan un *Access Point* en su casa o lugar de trabajo, tal y como sale de la caja, configurando su portátil, PDA o celular de la forma más rápida y cómoda (sin ningún tipo de encriptación).

6.1. Wardriving

Se llama *Wardriving* a la técnica de recorrer y relevar sitios o lugares en búsqueda de *redes wireless*. Este recorrido se realiza básicamente con una laptop equipada con por lo menos una placa de red wireless capaz de funcionar en modo monitor (también llamado modo promiscuo)⁶⁰, una serie de herramientas de análisis de tráfico (como *Airsnort* o *Kismet*), en ocasiones un equipo de GPS, y generalmente montado sobre algún vehículo con el que se recorrerán las calles de la ciudad.

6.1.1 Relevamientos publicados

Estos relevamientos permiten generar estadísticas a nivel mundial, como las realizadas durante el World Wide War Drive 4[WWWDrv] el cual indicó que de los 228.537 Access Point encontrados en una semana, más del 60% de ellos no tenía ningún tipo de cifrado[cybWar2007], o el realizado todos los años por la empresa CyberSec S.A.⁶¹ a partir del wardriving realizado en la zona céntrica de la ciudad de Buenos Aires en agosto de 2007[cybWar2007], que indica los siguientes resultados:

⁶⁰ Placas soportadas: http://www.wildpackets.com/support/product_support/airopeek/hardware

⁶¹ CyberSec [www.cybsec.com] es una empresa Argentina que desde 1996 se dedica exclusivamente a prestar servicios profesionales especializados en Seguridad Informática y desde hace cuatro años realiza wardriving en Buenos Aires y publica todos los años los resultados obtenidos.

- Total de redes descubiertas: 8357
- Total de redes con WPA: 1662 (19,47 %)
- Total de redes con WPA2: 394 (4,71 %)
- Total de redes con WEP: 2798 (33,48 %)
- Total de redes sin cifrado: 3501 (41,89 %)
- Total de redes por defecto: 117 (1,40 %)
- Total de redes ocultas: 108 (1,29 %)
- Cantidad total de clientes: 5875

Del mismo wardriving, se extrajo el siguiente mapa con un relevamiento de redes wireless sin encriptar en la ciudad de Buenos Aires.



6.2. Seguridad en las comunicaciones inalámbricas.

El estándar 802.11 establece diversos mecanismos para conseguir entornos seguros. Los más habituales son:

- Wired Equivalent Protocol (WEP).
- Wired Equivalent Protocol 2 (WEP2).
- Open System Authentication
- Access Control List (ACL)
- Closed Network Access Control

6.2.1 Método de cifrado WEP

WEP, acrónimo de Wired Equivalent Privacy, fue el primer protocolo de encriptación introducido en el primer estándar IEEE 802.11 en el año 1999. Está basado en el algoritmo de encriptación RC4, con una clave de 40 bits, combinada con un vector de inicialización (a partir de ahora IV) de 24 bits para encriptar el mensaje, y su correspondiente checksum (Integrity Check Value, a partir de ahora ICV). También existen versiones de este algoritmo con claves de 128bits y de 256bits.

Mensaje Encriptado = [Texto + checksum(Texto)] XOR [RC4(Clave + Vector de Inicialización)]

WEP es inseguro debido a su arquitectura, por lo que aumentar el tamaño de las claves de encriptación solo retrasaría el tiempo necesario para romperlo, en donde en el peor de los casos no superaría los 5 minutos. A continuación se enumeran y definen las deficiencias más reconocidas en la encriptación WEP.

6.2.2 Deficiencias de la encriptación WEP

El CRC32 o código de redundancia cíclica es el sistema utilizado para detectar si la información, en una transición de datos, está corrupta o dañada. Este método utiliza un polinomio generador $G(x)$ de grado r con n bits de datos binarios (coeficiente del polinomio de orden $n-1$). A estos bits de datos se le añaden r bits de redundancia, de forma que el polinomio resultante sea divisible por el polinomio generador. El receptor verificará si el polinomio recibido es divisible por $G(x)$ para asegurar que la transmisión fue correcta. El polinomio generador empleado por el comité de estándares de redes locales (IEEE-802) es el siguiente:

CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

En el año 2001, un grupo de investigadores de la Universidad de California, en Berkeley, hicieron pública una vulnerabilidad, donde se demostraba que era posible interceptar y alterar transmisiones wireless que se apoyaran sobre el algoritmo WEP ya que el campo ICV (Integrity Check Value) es generado simplemente haciendo un CRC de 32 bits del payload de la trama, el cual contiene dos serios problemas. En primer lugar, los CRC son independientes de la llave utilizada y del vector de inicialización. Y en segundo lugar, se conoce que los CRCs son lineales, lo que permite generar un ICV

válido ya que el CRC se combina con una operación XOR que también es lineal, lo que permite hacer “*bit flipping*” modificando el mensaje y el ICV de forma conocida, y así poder armar un paquete válido modificado.

Otra de las deficiencias del protocolo está dada por el tamaño del campo del vector de inicialización, el cual tiene solo 24 bits de longitud sin encriptar. Matemáticamente demostrable es que esto equivale a 16 millones de paquetes generables en muy poco tiempo sobre una red con un tráfico constante. Esto genera que el vector de inicialización se repita frecuentemente lo cual permite la posibilidad de realizar ataques estadísticos donde dos tramas son encriptadas con el mismo vector de inicialización.

6.2.3 Prueba de laboratorio

Para comprobar la facilidad con que se pueden obtener las contraseñas de una *red wireless* cifrada con un método blando, se armó una red inalámbrica con un *Access Point* marca *Encore*, configurado con un método de encriptación WEP y una contraseña de ingreso al anillo de catorce caracteres alfanuméricos; y dos *Mac Mini* en el laboratorio de sistemas de la UNLu. A su vez, se utilizó una laptop *Compaq v3218la*, con una placa de red wireless *Broadcom BCM94311MCG* con un sistema operativo *Ubuntu Festy* y las herramientas instaladas para realizar las capturas y el análisis de las mismas.

Se generaron archivos *.cap* con las capturas logradas detectando paquetes enviados desde una *MacMini* a la otra, utilizando la placa wireless en modo monitor.

Como se puede ver a continuación, con un ataque al archivo *eth1-06.cap*, de aproximadamente 350 megabytes, el *aircrack* tardó solo 1 segundo para retornar en texto claro la clave utilizada.

```
root@pablo-laptop:~# aircrack-ng -a 1 -c /home/pablo/capturas/eth1-06.cap
Opening /home/pablo/capturas/eth1-06.cap
Read 530204 packets.
# BSSID          ESSID          Encryption
1 00:18:E7:14:8D:23 ESSID-privada WEP (261566 IVs)
Choosing first network as target.
                                Aircrack-ng 0.9
                                [00:00:01] Tested 54 keys (got 261566 Ivs)
KB   depth  byte(vote)
0    0/ 1    6C( 48) 2A( 12) 42( 6) 20( 0) 22( 0) 25( 0) 26( 0) 29( 0) 2F( 0) 30( 0)
1    0/ 1    34( 52) 6A( 14) 7B( 4) 32( 3) 43( 3) 59( 3) 76( 3) 69( 2) 29( 0) 33( 0)
2    0/ 2    76( 58) 31( 36) 43( 12) 70( 12) 2E( 6) 3C( 6) 47( 5) 48( 5) 4C( 5) 59( 5)
```

```
3  0/ 1  69( 77) 3F(  6) 68(  5) 30(  4) 7B(  4) 00(  0) 21(  0) 43(  0) 44(  0) 49(  0)
4  0/ 1  64( 42) 50( 20) 46( 17) 7A( 15) 45( 13) 42(  6) 51(  6) 5E(  6) 60(  6) 57(  5)
5  0/ 1  34( 95) 32( 15) 6D( 13) 29(  8) 24(  6) 28(  6) 2F(  6) 40(  6) 54(  6) 21(  3)
6  0/ 1  65( 99) 2A( 18) 45( 12) 68( 11) 5E( 10) 26(  8) 27(  5) 29(  5) 4F(  5) 61(  5)
7  0/ 1  73( 51) 2A( 20) 70( 15) 7D( 15) 2B( 14) 67( 12) 44( 10) 65( 10) 38(  9) 39(  9)
8  0/ 1  62( 85) 4C( 36) 5C( 13) 71(  8) 3A(  5) 3C(  5) 73(  5) 75(  5) 3B(  4) 6E(  3)
9  0/ 2  65( 48) 52( 27) 5A( 23) 3C( 12) 73( 10) 5C(  8) 7C(  8) 3F(  6) 4C(  6) 4E(  6)
10 0/ 1  6C(105) 28( 13) 25(  8) 23(  6) 20(  5) 36(  5) 59(  5) 5E(  5) 22(  3) 33(  3)
11 0/ 1  6C(319) 23( 55) 3D( 50) 6D( 40) 60( 37) 6B( 36) 68( 35) 76( 35) 6E( 34) 4F( 33)
```

```
KEY FOUND! [ 6C:34:76:69:64:34:65:73:62:65:6C:6C:34 ] (ASCII: C0ntr4s3n14s )
```

```
Decrypted correctly: 100%
```

```
root@pablo-laptop:~#
```

6.2.4 Debilidades del algoritmo Key Scheduling de RC4

Scott fluhrer, Itsik Mantin y Adi Shamir demostraron que utilizando sólo la primera palabra de un *keystream*, podían obtener información de la clave secreta compartida. Se buscan *Ivs* que causen ausencia de información de la clave en el *keystream*. A este suceso se lo denominó “*resolved condition*” o condición resuelta. Cada uno de estos paquetes solo tiene ausencia de un byte de la clave, y este byte debe ser adivinado exitosamente para que el próximo paquete pueda ofrecer información del siguiente byte. *Wepcrack*⁶² y *Airsnort*⁶³ aprovechan esta debilidad para realizar ataques sobre redes WEP. En el próximo capítulo se verá un caso de uso de este tipo de software donde se podrán analizar los tiempos sobre una red wireless securizada con encriptación WEP.

6.2.5 Método de cifrado WPA

Para solucionar las limitaciones de seguridad que proporciona WEP, se recurrió a un subconjunto de la especificación 802.11i, el estándar de seguridad de las redes wireless, la cual será la versión definitiva cuando sea aprobada. La información sigue siendo cifrada utilizando el algoritmo RC4, con clave de 128 bits y un vector de inicialización de 48 bits, pero utiliza TKIP (*Temporal Key Integrity Protocol*), un protocolo que evita la reutilización de claves (una de las fallas de WEP) y agrega a esto, MIC (Message Integrity Check) para reforzar la verificación de integridad de los datos.

⁶² Wepcrack

⁶³ Airsnort

6.3. 802.11i

Es un WPA con “esteroides”, también llamado WPA2 posee mayor seguridad que su antecesor al implementar el cambio esperado reemplazando el algoritmo de encriptado RC4 (utilizados en WEP y WPA) por uno mucho más avanzado, el AES (o Advanced Encryption Standard). Este posee dos versiones, una personal que controla el acceso utilizando PSK (Pre-Shared Key), y la otra empresarial que utiliza la verificación de usuarios implementando el protocolo 802.1X que ofrece un proceso de variación dinámica de claves ajustado con otro protocolo denominado EAP (Extensible Authentication Protocol) donde el cliente se encuentra autenticado con una clave única que se va modificando automáticamente y es continuamente negociada por el servidor de manera transparente al usuario.

6.4. Consideraciones generales

A continuación se enumeran algunas consideraciones generales a la hora de armar una red wireless.

6.4.1 Ocular ESSId y Beacon frames

Cada red wireless se identifica con un *ESSId* (*Extended Service Set Identifier*) formado por una combinación de hasta 32 caracteres. Para poder ser parte de la red, el *ESSId* del *Access Point* y el *ESSId* del dispositivo móvil deben concordar. A su vez, los *Access Point* envían constantemente mensajes a los clientes de la red para que estos puedan detectar su presencia y así poder establecer el vínculo. Estos mensajes se denominan Beacon Frames y en ellos viajan los recientemente nombrados *ESSId*. Esta característica de los Access Point es conveniente que se encuentre deshabilitada o configurada de forma que los *ESSId* no sean emitidos en los mismos.

6.4.2 ACLs basadas en MACs

En la configuración de los Access Point también es posible y conveniente establecer *ACLs* o políticas de acceso en las cuales se deniegue todo acceso a la red excepto a aquellos clientes con las direcciones *MAC* conocidas por el *Access Point*. Estas direcciones *MAC* son códigos que utilizan los fabricantes de placas de red para identificarlas de manera única. Este tipo de filtrado sería perfecto si no fuera que en la práctica es muy fácil cambiar esta *MAC* desde un linux con la instrucción *ifconfig*, que a

su vez, combinado con el buen uso de un sniffer como por ejemplo el *Ethereal* o *Wireshark* se puede observar que en realidad esta práctica de *ACLs* por *MAC* solo retrasaría la intrusión.

6.4.3 Acotar el espacio de red

Además de activar el filtrado por MACs, se recomienda establecer el número máximo de dispositivos que puedan conectarse al mismo tiempo al Access Point. A su vez, es aconsejable desactivar el DHCP (Dynamic Host Configuration Protocol) y fijar la dirección IP y la puerta de enlace de forma manual.

6.5. Conclusión

Hoy en día encontramos dispositivos móviles que operan con redes wireless por todos lados, cualquier computadora portátil, PDA o celular incluyen esta nueva tecnología y muchos de los individuos que adquieren estos equipos suelen querer aprovechar todas las características de los mismos instalando un Access Point como sale de la caja. Como se comprobó con los router wireless *LINKSYS*, los cuales vienen con un CD de instalación y etiquetas en todo el equipo que indican “*no retirarlas hasta haber instalado el disco*”. El mismo que sirve de guía para realizar una instalación exitosa en siete pasos, sin que siquiera se nombre la palabra seguridad.

Por lo tanto, si no conocemos las vulnerabilidades de este tipo de redes, es imposible que seamos capaces de protegerlas de forma correcta.

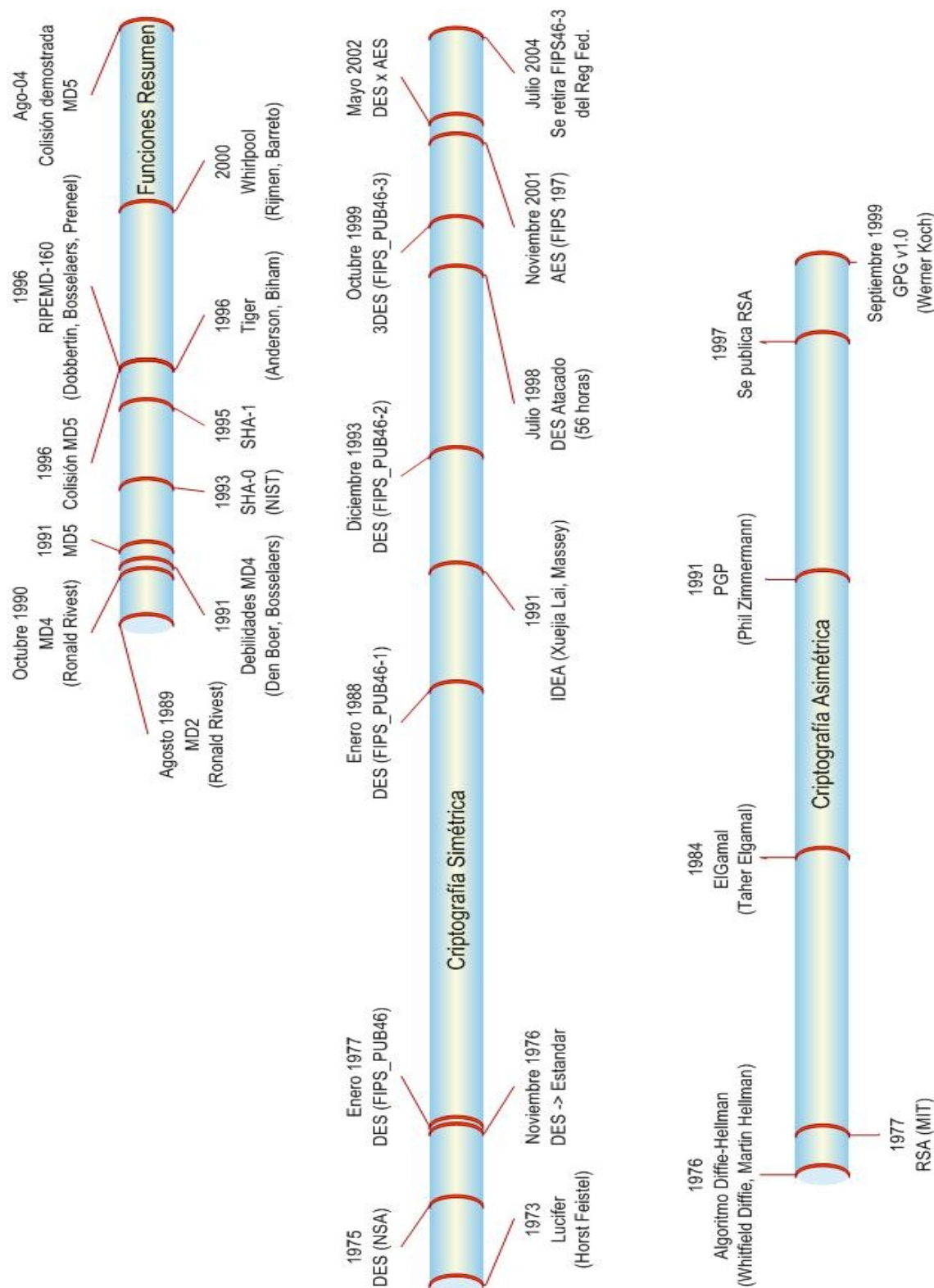
7. Conclusión General

Los sistemas operativos descriptos en este documento derivan de uno de los más poderosos de todos los tiempos, el Unix. Dichos sistemas son una de las bases para obtener plataformas seguras que corran todo tipo de servicios. Pero la responsabilidad de una plataforma segura no cae únicamente sobre el sistema operativo en el cual se apoya; sino que debe considerarse una posición compartida, junto con el administrador del sistema y los usuarios finales. El administrador es quién mantiene la potestad de decidir que medidas se deben tomar para alcanzar un nivel de seguridad óptimo, y también quién debe garantizar la optimización del sistema día a día para que, con el pasar del tiempo, no se convierta en un sistema obsoleto. El papel de los usuarios nos es menos importante, ya que deberían estar educados para que asuman su papel como elemento activo en el sostenimiento de las políticas de seguridad definidas. En cuanto a dichas medidas de seguridad, nunca se llegará al momento en que esté todo escrito y dicho sobre el tema, sino que se convertirá en un camino que crece paralelamente con el avance tecnológico de las comunicaciones y el manejo de la información.

Para lograr estas premisas, no es necesario que el administrador sea un experto en la materia, sino que sea capaz de comprender el funcionamiento del sistema y tener la capacidad de reconocer con criterio las medidas a tomar para mantener el control de la situación. Dicha capacidad se logra con un conjunto de características donde no puede faltar la incorporación de información al día sobre las fallas encontradas (ver Anexo V) y las actualizaciones existentes, la habilidad de poner en práctica nuevos conceptos personales y la implementación constante de nuevas técnicas de vigilancia para mantenerse preparados ante un posible nuevo ataque.

La cultura de la libertad del conocimiento hace posible que cualquier persona tenga la capacidad de acceso a dicha información, disponible en libros electrónicos, trabajos de investigación, sitios webs dedicados y blogs de administradores que comparten sus experiencias con el resto de la comunidad, y es por eso que la solución somos cada uno de nosotros, parte importante de este sistema seguro.

8. Anexo I – Línea de tiempo de Evolución de Sistemas Criptográficos.



9. Anexo II – Técnicas de evasión de Portsentry

Para concretar esta práctica se conectaron dos máquinas reales, una con un sistema operativo Windows XP SP2 y otra con un Linux Ubuntu Festy Desktop, la cual ejecutaba una máquina virtual con un Ubuntu Festy Server.

- Zombie: Windows XP SP2 – Red: 10.218.106.63/24
- Atacante: Ubuntu Server 7.10 – Red: 10.218.106.60/24 – Nmap 4.60
- Defensa: Ubuntu Desktop 7.10 – Red: 10.218.106.64/24 – Portsentry 1.2

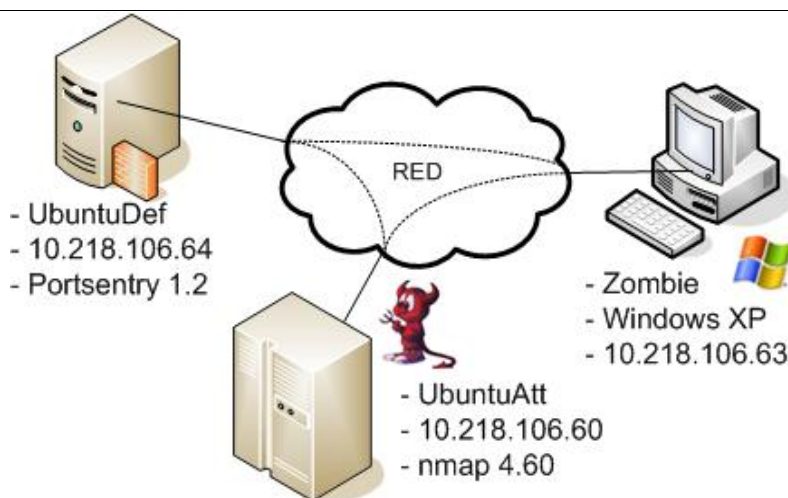


Figura 9.1 – Esquema de red.

En la máquina de ataque se verificó que exista una computadora con al menos un puerto abierto y uno cerrado para poder utilizar como zombie.

```
root@ubuntuAtt:~# nmap -P0 10.218.106.63
Starting Nmap 4.60 ( http://nmap.org ) at 2008-04-26 12:36 ART
Interesting ports on winXPsp2 (10.218.106.63):
Not shown: 1707 closed ports
PORT      STATE SERVICE
80/tcp    open  http
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
1680/tcp   open  CarbonCopy
3306/tcp   open  mysql
3389/tcp   open  ms-term-serv
8081/tcp   open  blackice-icecap
MAC Address: 00:02:E3:51:33:C4 (Lite-on Communications)
Nmap done: 1 IP address (1 host up) scanned in 1.994 seconds
root@ubuntuAtt:~#
```

Desde la máquina de ataque se realizó un escaneo de puertos en modo idle [NMAP_IDLE] a través de la máquina zombie (se acotó el rango de Ips para acelerar las capturas y simplificar lecturas de logs):

```
root@ubuntuAtt:~# nmap -P0 -p15-25 -sI 10.218.106.63 10.218.106.64
Starting Nmap 4.60 ( http://nmap.org ) at 2008-04-26 14:30 CST
Idle scan using zombie 10.218.106.63 (10.218.106.63:80); Class: Incremental
Interesting ports on 10.218.106.64:
PORT      STATE      SERVICE
15/tcp    closed|filtered netstat
16/tcp    closed|filtered unknown
17/tcp    closed|filtered qotd
18/tcp    closed|filtered msp
19/tcp    closed|filtered chargen
20/tcp    closed|filtered ftp-data
21/tcp    closed|filtered ftp
22/tcp    open       ssh
23/tcp    closed|filtered telnet
24/tcp    closed|filtered priv-mail
25/tcp    closed|filtered smtp
MAC Address: 00:16:D3:9E:DD:64 (Wistrin)
Nmap done: 1 IP address (1 host up) scanned in 2.241 seconds
root@ubuntuAtt:~#
```

En la máquina de defensa se activó el Portsentry y verificó el archivo de log mediante el comando *tail*.

```
root@ubuntuDef:~# tail -f /var/log/syslog | grep portsentry
Apr 26 12:22:47 ubuntuDef portsentry[31751]: attackalert: TCP SYN/Normal scan from host:
10.218.106.63/10.218.106.63 to TCP port: 15
Apr 26 12:22:47 ubuntuDef portsentry[31751]: attackalert: Ignoring TCP response per
configuration file setting.
Apr 26 12:22:47 ubuntuDef portsentry[31751]: attackalert: TCP SYN/Normal scan from host:
10.218.106.63/10.218.106.63 to TCP port: 15
Apr 26 12:22:47 ubuntuDef portsentry[31751]: attackalert: Host:
10.218.106.63/10.218.106.63 is already blocked Ignoring
root@ubuntuDef:~#
```

Como se pudo observar, el Portsentry detectó el ataque pero desde la ip incorrecta, la máquina utilizada como zombie. No fué necesario, pero si el Portsentry hubiese estado en modo reactivo habría bloqueado otra IP. La misma prueba fué realizada utilizando internet⁶⁴ en lugar de una red local dando los mismos resultados.

⁶⁴ Se realizó un escaneo normal entre dos Ips conocidas de la red del proveedor Speedy, encontrando al instante varias máquinas que podían ser usadas como zombies. Mediante una de ellas se atacó la propia con el Portsentry activado.

10. Anexo III - Formato de encabezado de paquetes

Para poder establecer una línea de defensa basada en reglas (ya sea para la configuración de un Ids o un Firewall) es indispensable conocer el manejo de los encabezados de cada uno de los protocolos.

10.1. Encabezado de paquetes IP

El encabezado básico de paquetes *Ipv4* está formado por 20 bytes que puede ser extendido opcionalmente con hasta 40 bytes de longitud en un campo de opciones. El detalle de la descripción de este encabezado puede ser encontrado en la RFC791 disponible en <ftp://ftp.isi.edu/in-notes/rfc791.txt> y se resume en la siguiente descripción:

- **Versión (4 bits):** Define el número de versión de cabecera utilizada. Para la actual es común encontrar un cuatro en este campo (0100).
- **Tamaño de Cabecera (4 bits):** Longitud en palabras de 32bits .
- **Tipo de Servicio (8 bits):** Indica una serie de parámetros sobre la calidad de servicio. Los 5 bits de menor peso indican las características del servicio:
 - Bit 0: Debe permanecer en 0.
 - Bit 1: Costo
 - 1 para mínimo.
 - 0 para normal.
 - Bit 2: Fiabilidad
 - 1 para máxima.
 - 0 para normal.
 - Bit 3: Rendimiento
 - 1 para máximo.
 - 0 para normal.
 - Bit 4: Retardo
 - 1 para minimo.
 - 0 para normal.
 - Bits 5, 6 y 7: Indican el nivel de urgencia basado en el sistema militar de precedencia de la CCEB⁶⁵.

⁶⁵ Organización de comunicaciones electrónicas militares formada por 5 naciones la cual desarrolló el Mesage Precedence (http://en.wikipedia.org/wiki/Message_precedence)

- 000: De rutina.
- 001: Prioritario.
- 010: Inmediato.
- 011: Relámpago.
- 100: Invalidación relámpago.
- 101: Procesando llamada crítica y de emergencia.
- 110: Control de trabajo de Internet.
- 111: Control de red.
- **Longitud Total (16 bits):** Tamaño total en octetos del datagrama (cabecera más datos). En caso de fragmentación, contendrá el tamaño de la parte, no del datagrama original.
- **Identificador (16 bits):** Identificador único del datagrama, utilizado en caso de que el mismo deba ser fragmentado para poder distinguir las partes de un datagrama de los de otro.
- **Indicadores (3 bits):** Utilizado para especificar valores relativos a la fragmentación de paquetes.
 - Bit 0: Reservado, debe ser 0.
 - Bit 1:
 - 0 para divisible.
 - 1 para no divisible.
 - Bit 2:
 - 0 para último fragmento.
 - 1 para fragmento intermedio.
- **Posición del fragmento (13 bits):** Indica la posición en unidades de 64 bits que ocupa el paquete actual dentro del datagrama original (0 para el primer paquete).
- **Tiempo de vida (8 bits):** Número máximo de enrutadores que un paquete puede atravesar. En cada uno de los mismos, este número es disminuido en uno hasta que llegue al valor 0. En tal caso el paquete no será reenviado.
- **Protocolo (8 bits):** Indica el protocolo utilizado en el siguiente nivel para la parte de datos del datagrama. Según la lista de protocolos de la *IANA*⁶⁶ se pueden destacar los siguientes valores⁶⁷:

⁶⁶ Internet Numbers Authority o Agencia de Autorización de Números de Internet sustituida por la ICANN (Internet Corporation for Assigned Names and Numbers o Corporación de Internet para la Asignación de Nombres y Números) a partir de septiembre de 1998.

⁶⁷ Lista completa disponible en <http://support.microsoft.com/kb/289892>

- 00000001 para ICMP.
- 00000010 para IGMP.
- 00000110 para TCP.
- 00001000 para EGP.
- 00001001 para IGP
- 00010001 para UDP.
- 00101001 para Ipv6 (encapsulado).
- **Checksum de la cabecera (16 bits).**
- **Dirección IP de Origen (32 bits).**
- **Dirección IP de Destino (32 bits).**
- **Opciones (variable):** Aunque no es obligatoria, cualquier ruteador tiene que poseer la capacidad de interpretación del campo. Puede tener dos formatos:
 - Simple:
 - Bit 0: Para el caso de fragmentación:
 - 0 si no se copia la opción.
 - 1 si se copia la opción.
 - Bit 1 y 2:
 - 00 para control.
 - 01 para reservada.
 - 10 para depuración y mediciones.
 - 11 reservada.
 - Bit 3 al 7: Se utilizan para identificador de la opción.
 - Compuesto:
 - Tipo de Opción:
 - Tamaño de Opción (Incluye el Tipo, Tamaño y Datos).
 - Datos de Opción.
- **Relleno (Variable):** Utilizado para asegurar un tamaño de la cabecera (en bits) múltiplo de 32 con 0 por defecto.

0-3	4-7	8-15	16-18	19-31
Versión	Tamaño Cabecera	Tipo de Servicio	Longitud Total	
Identificador			Indicadores	Posición de Fragmento
Time To Live		Protocolo	Checksum Cabecera	
Dirección IP de Origen				
Dirección IP de Destino				
Opciones				Relleno

Figura 9.1 – Cabecera IP

10.2. Encabezado ICMP

El encabezado de este protocolo se encuentra completamente definido en la RFC 792 disponible en <ftp://ftp.isi.edu/in-notes/rfc792.txt>, y muestra la estructura que se describe a continuación:

- **Tipo (8 bits).**
- **Código (8 bits):** Código del subtipo utilizado.
- **Checksum (16 bits):** Utilizados para detectar errores en el paquete.
- **Datos (variable/opcional).**

	Bits 160-167	168-175	176-183	184-191
160	Type	Code	Checksum	
192	ID		Sequence	

Figura 9.2 – Cabecera Básica ICMP

10.2.1 Lista (Tipo/Código) de mensajes de control más comunes:

- 0 - Echo Reply
 - 0 Echo reply (usado en ping)
- 3 - Destination Unreachable
 - 0 Destination network unreachable
 - 1 Destination host unreachable
 - 2 Destination protocol unreachable
 - 3 Destination port unreachable
 - 4 Fragmentation required, and DF set
 - 5 Source route failed
 - 6 Destination network unknown
 - 7 Destination host unknown
 - 8 Source host isolated
 - 9 Network administratively prohibited
 - 10 Host administratively prohibited
 - 11 Network unreachable for TOS
 - 12 Host unreachable for TOS
 - 13 Communication administratively prohibited
- 8 - Echo Request
 - 0 Echo request

- 9 - Router Advertisement
 - 0 Router Advertisement
- 10 - Router Solicitation
 - 0 Router discovery/selection/solicitation
- 11 - Time Exceeded
 - 0 TTL expired in transit
 - 1 Fragment reassembly time exceeded
- 15 - Information Request
 - 0 Information Request
- 16 - Information Reply
 - 0 Information Reply
- 30 - Traceroute
 - 0 Information Request

10.2.2 ICMP Echo Request

El Echo Request es un protocolo ICMP que es enviado para que el host remoto conteste un Echo Reply. Al encabezado original de ICMP se agregan los datos de identificador (16 bits), número de secuencia (16 bits).

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31								
Tipo = 8								Código = 0								Checksum																							
Identificador																Número de secuencia																							
Datos :::																																							

Figura 9.3 – Cabecera ICMP Echo Request

10.2.3 ICMP ECHO Reply

El Echo Reply es un mensaje de respuesta generado a partir de un Echo Request . Además de los datos de identificación y secuencia con los que se puede asociar cada Echo Reply a los Echo Request, se incluye el campo *datos* tal cual es recibido.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31								
Tipo = 0								Código = 0								Checksum																							
Identificador																Número de secuencia																							
Datos :::																																							

Figura 9.4 – Cabecera ICMP Echo Reply

10.2.4 Encabezado TCP

TCP es un protocolo de comunicación orientado a conexión y confiable, actualmente documentado por IETF en la RFC 793 disponible en <ftp://ftp.isi.edu/in-notes/rfc793.txt>. A continuación se detallan sus campos:

- Puerto Origen (16 bits).
- Puerto Destino (16 bits).
- Número de Secuencia (32 bits).
- Número de Acuse de Recibo o ACK (32 bits).
- Longitud de Cabecera TCP o Offset (4 bits).
- Flags o Bits de control (8 bits): 1 indica que está seteado.
 - CWR: Congestion Window Reduced.
 - ECE: Echo.
 - URG: Flag de Urgente.
 - ACK: Acuse de recibo.
 - PSH: “Empujar” o priorizar el paquete.
 - RST: Final de conexión.
 - SYN: Solicitud de Sincronización.
 - FIN: El emisor ha llegado al final de su flujo de octetos.
- Ventana (16 bits): Tamaño de la ventana que el receptor se encuentra dispuesto a recibir.
- Checksum (16 bits).
- Puntero Urgente (16 bits): Usado solo si el flag URG está activado.
- Opciones (Variable)
- Datos (Variable)

+	Bits 0 - 3	4 - 7	8 - 15	16 - 31
0	Puerto Origen			Puerto Destino
32	Número de Secuencia			
64	Número de Acuse de Recibo (ACK)			
96	longitud cabecera TCP	Reservado	Flags	Ventana
128	Suma de Verificación (Checksum)			Puntero Urgente
160	Opciones + Relleno (opcional)			
224	Datos			

Figura 9.5 – Segmento TCP

10.3. Encabezado UDP

El protocolo UDP permite el envío de datagramas sin necesidad de establecer una conexión. No posee confirmación ni control de flujo. Sus usos más conocidos pueden ser encontrados en la implementación de DHCP y DNS. Su encabezado se describe en la RFC 768 y puede ser obtenido de <http://www.ietf.org/rfc/rfc768.txt>.

- **Puerto Origen (16 bits):** Campo opcional que identifica el puerto del emisor.
- **Puerto Destino (16 bits).**
- **Longitud del Mensaje (16 bits):** Tamaño en bytes del datagrama UDP incluido el encabezado.
- **Checksum (16 bits):** Opcional en teoría, aunque generalmente se utiliza. Es una suma de verificación que abarca la cabecera, los datos y una pseudo-cabecera con las Ips de origen/destino, protocolo, longitud del datagrama y ceros hasta completar un múltiplo de 16.

+	Bits 0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

Figura 9.6 – Protocolo UDP

10.4. Encabezado ARP

El ARP o Address Resolution Protocol (Protocolo de resolución de direcciones) es un protocolo a nivel de red responsable de encontrar la dirección MAC correspondiente a determinada IP. La estructura de sus encabezados se describen a continuación:

- **Tipo de Hardware (16 bits):** A cada protocolo de la capa de enlace se le asigna un número, por ejemplo ethernet es 1.
- **Tipo de Protocolo (16 bits):** Al igual que el caso anterior, a cada protocolo se le asigna una identificación. Por ejemplo Ipv4 es 0x0800.

- **Tamaño de la dirección de Hardware (8 bits):** Tamaño en bytes de la dirección de hardware. Para ethernet son 6 bytes.
- **Tamaño del Protocolo (8 bits):** También representados en bytes, para el caso de Ipv4 son 4 bytes.
- **Operación (16 bits):** Especifica la operación del remitente.
 - 1 para request.
 - 2 para reply.
- **Dirección de hardware del origen (variable):** Para el caso de ethernet son 48 bits.
- **Protocolo del origen (variable):** Para el caso de paquetes IPv4 son 32 bits.
- **Dirección de hardware destino (variable):** Para el caso de ethernet son 48 bits.
- **Protocolo destino (variable):** Para el caso de paquetes IPv4 son 32 bits.

+	Bits 0 - 7	8 - 15	16 - 31
0	Hardware type (HTYPE)		Protocol type (PTYPE)
32	Hardware length (HLEN)	Protocol length (PLEN)	Operation (OPER)
64	Sender hardware address (SHA)		
?	Sender protocol address (SPA)		
?	Target hardware address (THA)		
?	Target protocol address (TPA)		

Figura 9.7 – Encabezado ARP

11. Anexo IV – Técnicas de evasión para la detección remota del SO.

Para realizar estas pruebas se montó una red virtual con una máquina *vmWare* con *openBSD* 4.2 tal y como viene recién instalado (con el *pf* deshabilitado).

- *vmWare* player con *openBSD* 4.2 (64 bits) – Red 10.218.106.100/24
- Laptop con Linux Ubuntu Gutsy 7.10 (64 bits) – *nmap* 4.53 – Red 10.218.106.64/24

El resultado después de realizar un ataque con el *pf* desactivado indicó inmediatamente la versión del sistema operativo al detalle, el IPID está establecido como *random* en su configuración por default [FINGER06].

```
root@pablo-laptop:/# nmap -v -A 10.218.106.100

Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-29 12:07 ART
Initiating ARP Ping Scan at 12:07
Scanning 10.218.106.100 [1 port]
Completed ARP Ping Scan at 12:07, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:07
Completed Parallel DNS resolution of 1 host. at 12:07, 0.03s elapsed
Initiating SYN Stealth Scan at 12:07
Scanning 10.218.106.100 [1714 ports]
Discovered open port 113/tcp on 10.218.106.100
Increasing send delay for 10.218.106.100 from 0 to 5 due to 44 out of 145 dropped probes
since last increase.
Discovered open port 22/tcp on 10.218.106.100
Discovered open port 13/tcp on 10.218.106.100
Discovered open port 37/tcp on 10.218.106.100
Completed SYN Stealth Scan at 12:07, 16.55s elapsed (1714 total ports)
Initiating Service scan at 12:07
Scanning 4 services on 10.218.106.100
Completed Service scan at 12:07, 6.12s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against 10.218.106.100
Retrying OS detection (try #2) against 10.218.106.100
SCRIPT ENGINE: Initiating script scanning.
Host 10.218.106.100 appears to be up ... good.
Interesting ports on 10.218.106.100:
Not shown: 1710 closed ports
PORT      STATE SERVICE VERSION
13/tcp    open  daytime
22/tcp    open  ssh      OpenSSH 4.7 (protocol 2.0)
37/tcp    open  time      (32 bits)
113/tcp   open  ident
MAC Address: 00:0C:29:13:EE:16 (VMware)
```

```
Device type: general purpose
Running: OpenBSD 3.X|4.X
OS details: OpenBSD 3.9 - 4.2
Uptime: 0.000 days (since Sat Mar 29 12:07:26 2008)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=263 (Good luck!)
IP ID Sequence Generation: Randomized

Read data files from: /usr/share/nmap
OS and Service detection performed. Please report any incorrect results at
http://insecure.org/nmap/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.978 seconds
      Raw packets sent: 1971 (88.150KB) | Rcvd: 1747 (70.766KB)

root@pablo-laptop:/#
```

Luego de habilitar el pf en el archivo /etc/rc.conf y hacer unas mínimas configuraciones en el /etc/pf.conf el sistema operativo verificado con el mismo método quedó completamente oculto.

en /etc/rc.conf

```
pf=YES
pf_rules=/etc/pf.conf
```

en /etc/pf.conf

```
int_if = "em0"
scrub in on $int_if all no-df random-id

pfctl -d                                # para deshabilitar el pf
pfctl -f /etc/pf.conf  # para recargar la configuración del pf
pfctl -e                                # para habilitar el pf

#####
con no-df y random-id
#####
```

```
root@pablo-laptop:/# nmap -v -A 10.218.106.100
```

```
Starting Nmap 4.53 ( http://insecure.org ) at 2008-03-29 12:02 ART
Initiating ARP Ping Scan at 12:02
Scanning 10.218.106.100 [1 port]
Completed ARP Ping Scan at 12:02, 0.02s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:02
Completed Parallel DNS resolution of 1 host. at 12:02, 0.03s elapsed
Initiating SYN Stealth Scan at 12:02
Scanning 10.218.106.100 [1714 ports]
Discovered open port 22/tcp on 10.218.106.100
```

```
Discovered open port 113/tcp on 10.218.106.100
Increasing send delay for 10.218.106.100 from 0 to 5 due to 44 out of 146 dropped probes
since last increase.
Discovered open port 37/tcp on 10.218.106.100
Discovered open port 13/tcp on 10.218.106.100
Completed SYN Stealth Scan at 12:02, 14.93s elapsed (1714 total ports)
Initiating Service scan at 12:02
Scanning 4 services on 10.218.106.100
Completed Service scan at 12:02, 6.12s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against 10.218.106.100
Retrying OS detection (try #2) against 10.218.106.100
Retrying OS detection (try #3) against 10.218.106.100
Retrying OS detection (try #4) against 10.218.106.100
Retrying OS detection (try #5) against 10.218.106.100
SCRIPT ENGINE: Initiating script scanning.
Host 10.218.106.100 appears to be up ... good.
Interesting ports on 10.218.106.100:
Not shown: 1710 closed ports
PORT      STATE SERVICE VERSION
13/tcp    open  daytime
22/tcp    open  ssh      OpenSSH 4.7 (protocol 2.0)
37/tcp    open  time      (32 bits)
113/tcp   open  ident
MAC Address: 00:0C:29:13:EE:16 (VMware)
No exact OS matches for host (If you know what OS is running on it, see
http://insecure.org/nmap/submit/ ).
TCP/IP fingerprint:
OS: SCAN (V=4.53%D=3/29%OT=13%CT=1%CU=33321%PV=Y%DS=1%G=Y%M=000C29%TM=47EE5A2
OS:3%P=x86_64-unknown-linux-gnu) SEQ (SP=106%GCD=1%ISR=10B%TI=RD%II=RI%TS=22)
OS: SEQ (SP=104%GCD=1%ISR=109%TI=RD%II=RI%TS=21) SEQ (SP=100%GCD=1%ISR=10F%TI=R
OS:D%II=RI%TS=22) SEQ (SP=103%GCD=1%ISR=10E%TI=RD%II=RI%TS=21) SEQ (SP=101%GCD=
OS:1%ISR=10D%TI=RD%II=RI%TS=21) OPS (O1=M5B4NNSNW0NNT11%O2=M5B4NNSNW0NNT11%O3
OS:=M5B4NWN0NNT11%O4=M5B4NNSNW0NNT11%O5=M5B4NNSNW0NNT11%O6=M5B4NNSNNT11) WIN (
OS:W1=4000%W2=4000%W3=4000%W4=4000%W5=4000%W6=4000) ECN (R=Y%DF=Y%T=40%W=4000
OS:%O=M5B4NNSNW0%CC=N%Q=) T1 (R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=) T2 (R=N) T3 (R
OS:=N) T4 (R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=) T5 (R=Y%DF=Y%T=40%W=0%S=Z%
OS:A=S+%F=AR%O=%RD=0%Q=) T6 (R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=) T7 (R=N)
OS:U1 (R=Y%DF=N%T=FF%TOS=0%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD
OS:=G) IE (R=Y%DFI=N%T=FF%TOSI=S%CD=S%SI=S%DLI=S)
Uptime: 0.000 days (since Sat Mar 29 12:02:54 2008)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=257 (Good luck!)
IP ID Sequence Generation: Randomized

Read data files from: /usr/share/nmap
OS and Service detection performed. Please report any incorrect results at
http://insecure.org/nmap/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 34.697 seconds
Raw packets sent: 1932 (89.296KB) | Rcvd: 1780 (73.268KB)
```

12. Anexo V - Herramientas libres para seguridad Informática y de Redes⁶⁸

12.1. Spyware

1. **Ad-Aware SE Personal** [Windows] – Herramienta para remover spyware, el cual mantiene una base de datos de spywares conocidos y a su vez tiene la capacidad de identificar código malicioso para una mejor detección.
Site: http://www.lavasoftusa.com/products/ad-aware_se_personal.php
2. **Spyware Terminator** [Windows] – Monitorea todos los procesos que se encuentran corriendo en tiempo real y determina la integridad de ellos. Tiene la capacidad de manejar una cuarentena si algún spyware es encontrado.
Site: <http://www.spywareterminator.com>.
3. **Spybot** [Windows] - Search & Destroy – Utiliza técnicas de escaneo en profundidad para limpiar el sistema. Permite mantener una lista negra.
Site: <http://www.safer-networking.org/en/index.html>
4. **Winpooch** [Windows] – Trabaja en tiempo real interceptando los accesos de programas a las claves de registro o disco. También puede detectar troyanos y se puede combinar con el ClamWin para que funcione como antivirus.
Site: <http://winpooch.free.fr/page/home.php?lang=en&page=home>
5. **SpywareBlaster** [Windows] – Previene la instalación de spyware basados en ActiveX, bloquea cookies y restringe las acciones de los sitios web maliciosos.
Site: <http://www.javacoolsoftware.com/sbdownload.html>
6. **Bazooka Adware and Spyware Scanner** [Windows] – Escanea y detecta spyware, adware, troyanos, keyloggers.
Site: <http://www.kephyr.com/spywarescanner/library/index.phtml?source=bassindex>

⁶⁸ <http://www.itsecurity.com/features/103-free-security-apps-041607/>
<http://tecnoticias.info/tips-comentarios/seguridad-en-tecnologia-informatica/50-herramientas-top-de-seguridad.html>

7. **Windows Defender** [Windows] – Esta herramienta es la solución que proporciona Microsoft para proteger la computadora de spyware y popups.
Site: <http://www.microsoft.com/downloads/details.aspx?FamilyID=435bfce7-da2b-4a6a-afa4-f7f14e605a0d&DisplayLang=en>
8. **MalWhere** [Windows] – Muestra información detallada acerca de los procesos que corren en la computadora. A diferencia de la mayoría, no requiere actualizar la definición de los spywares.
Site: <http://www.malwhere.com>
9. **Trend Micro Anti-Spyware** [Windows] – Detector online de spyware. Incluye herramientas para remover los mismos.
Site: <http://us.trendmicro.com/us/products/personal/free-tools-and-services/index.html>
10. **Zebra scanner** [Mac] – Escanea archivos buscando código oculto y notifica al usuario si varía la integridad de los archivos. No posee un método para desinfectar los mismos.
Site: <http://home.wanadoo.nl/denheyer/webpaginas/ZebraDocumentation.html>

12.2. Antivirus

1. **McAfee AVERT Stinger** [Windows] – Herramienta usada para remover un virus o worm específico.
Site: <http://vil.nai.com/vil/stinger>
2. **PC Tools AntiVirus** [Windows] – Suit de escaneo de virus, el cual ofrece protección escaneando programas de Internet, adjuntos de emails y a su vez se pueden realizar búsquedas en el sector de arranque y correr limpiezas en tiempo real.
Site: <http://www.pctools.com/free-antivirus>
3. **Clam AntiVirus** [Linux] – Es la solución más utilizada en máquinas con sistemas operativos basados en Unix. También incluye escaneos en tiempo real y posee una base de datos de definiciones de virus la cual es actualizada varias

veces al día.

Site: <http://www.clamav.net>

4. **AVScan** [Linux] – Es un front-end del antivirus Clam.

Site: <http://wolfpack.twu.net/Endeavour2/contrib/index.html#avscan>

5. **ClamXav** [Mac] – Utiliza el motor del antivirus ClamAV para escanear archivos en busca de virus.

Site: <http://www.clamxav.com>

6. **AVG Anti-Virus** [Windows, Linux] – Otra efectiva suite de escaneo de virus. Requiere regulares actualizaciones en sus definiciones de virus.

Site: <http://free.grisoft.com/doc/2/lng/us/tpl/v5>

7. **Avast Home Edition** [Windows, Linux] – Al igual que la suit anterior, con esta herramienta se puede escanear en tiempo real, analizar emails, posee filtro de tráfico de Internet, y un firewall.

Site: <http://avast.com>

8. **Housecall** [Web: Windows, Linux, Mac] – Aplicación basada en web, que permite escanear virus y a su vez permite removerlos, como también a worms, troyanos y spyware.

Site: <http://housecall.trendmicro.com>

9. **Symantec Security Check** [Web] – Es una solución simple que realiza búsquedas en una computadora de malware, y a su vez permite su eliminación.

Site: <http://security.symantec.com/sscv6/default.asp?langid=ie&venid=sym>

12.3. Anti-Rootkit

1. **Sophos Anti-Rootkit** [Windows] – Remueve rootkits y varios tipos de malware que puedan correr ocultos en la red o de manera local.

Site: <http://www.sophos.com/products/free-tools/sophos-anti-rootkit.html>

2. **Rootkit Hunter** [Mac] – Busca y remueve los más populares rootkits de UNIX

y Mac.

Site: <http://www.christian-hornung.de>

3. **chkrootkit** [Linux] – Es la solución recomendada para encontrar y remover rootkits en máquinas Linux.

Site: <http://www.chkrootkit.org>

12.4. Firewall

1. **ZoneAlarm** [Windows] – Uno de los más populares firewalls para Windows que incluye un fácil método para de escribir reglas de seguridad.

Site: <http://www.zonealarm.com/store/content/company/products/znalm/freeDownload.jsp>

2. **VisualZone** [Windows] – Analizador de información de ZoneAlarm incluyendo tipos de intrusión y métodos de acceso.

Site: <http://www.visualizesoftware.com/visualzone/visualzone.htm>

3. **PC Tools Firewall Plus** [Windows] – Firewall para usuarios hogareños que monitorea conexiones de red y utiliza una simple serie de reglas para prevenir malware desde otras computadoras.

Site: <http://www.pctools.com/firewall>

4. **Firestarter** [Linux] – Firewall de tiempo real que monitorea todos los puertos abiertos las conexiones de red activas. Permite especificar reglas bastante estrictas y posee una interfaz de simple manejo.

Site: <http://www.fs-security.com>

5. **Firewall Builder** [Linux] – Completa herramienta para preparar un set de reglas y políticas de firewall basado en aplicaciones Unix conocidas como Iptables o Ipfilter.

Site: <http://www.fwbuilder.org>

6. **TuxGuardian** [Linux] – Verifica la integridad de las aplicaciones que tratan de obtener acceso a la red. Previniendo la instalación de virus, troyanos, spyware, etc. desde la red externa.

Site: <http://tuxguardian.sourceforge.net>

7. **HardWall Firewall** [Linux] – Scripts basado en reglas de Iptables que realiza la inspección detallada de paquetes filtrando los mismos para mantener la computadora libre de tráfico malicioso.

Site: <http://freshmeat.net/projects/hwfirewall>

8. **BullDog** [Linux] – Firawall de uso complejo para usuarios avanzados de Unix.

Site: <http://tanaya.net/BullDog/index.shtml>

9. **PeerGuardian** [Windows, Linux, Mac] – Bloquea acceso de IPs determinadas al interior de la red, para evitar especialmente el tráfico de P2P.

Site: <http://phoenixlabs.org/pg2>

12.5. Protección de Email

1. **Thunderbird** [Windows, Linux, Mac] – Seguro cliente de mail altamente configurable y adaptable. Cuenta con filtro antispam, protección ante phishing y capacidad de encriptación mediante pluggins.

Site: <http://www.mozilla.com/en-US/thunderbird>

2. **Smart sendmail filters** [Linux] – Colección de herramientas para bloquear y filtrar spam además de escanear adjuntos en busca de virus.

Site: <http://smfs.sourceforge.net/index.html>

3. **Sagator** [Linux] – Interfaz para Postfix, Sendmail y otros servidores de smtp que permiten implementar soluciones de antivirus y antispams conocidos.

Site: <http://www.salstar.sk/sagator>

4. **Tiger Envelopes** [Windows, Linux, Mac] – Herramienta para encriptar mensajes de emails que se integra con Outlook, Thunderbird, Mac Mail y Kmail.

Site: <http://www.tigerprivacy.com>

5. **Simple ASCII Email Encoder** [Mac] – Codifica la dirección de email con su

equivalente en código ASCII para prevenir que el mismo sea obtenido por servidores de spam.

Site: <http://www.closeknitsoftware.net/?page=emailencoder>

12.6. Utilidades Web

1. **WinSCP** [Windows] – Cliente de SFTP (Secure FTP) y SCP (Secure Copy) para transferir archivos a servidores. También incluye un editor de documentos remoto.

Site: <http://winscp.net/eng/index.php>

2. **Privacy Mantra** [Windows] – Borra los datos almacenados en el historial de Internet (cookies, cache, index.dat, formularios y más) para mantener la privacidad.

Site: <http://www.codecode.com/privacymantra>

3. **Tor** [Windows, Linux, Mac] – Arma un túnel virtual sobre la red para proveer anonimato en la navegación de Internet y la transferencia de archivos.

Site: <http://www.torproject.org/index.html.en>

4. **BarracudaDrive** [Windows, Linux, Mac] – Servidor web seguro libre, útil para el manejo de archivos en forma remota.

Site: <http://barracudaserver.com/products/HomeServer>

5. **Firefox** [Windows, Linux, Mac] – El *browser Open Source* más popular, de arquitectura segura y altamente adaptable por el uso (al igual que el *Thunderbird*) de *add-ons*.

Site: <http://www.mozilla.com/en-US/firefox>

6. **McAfee Site Advisor** [Windows, Linux, Mac] – Ranquea los sitios web por su capacidad de infección con programas maliciosos y advierte durante la navegación que tan seguro es un sitio web.

Site: <http://www.siteadvisor.com>

7. **Opera** [Windows, Linux, Mac] – Otro explorador de Internet seguro y

multiplataforma.

Site: <http://www.opera.com>

8. **PuTTY** [Windows, Linux] – Cliente de Telnet y SSH para establecer sesiones remotas. Es muy utilizado para conexión de máquinas Windows con Unix, práctico por no ser necesaria una instalación de la herramienta.

Site: <http://www.chiark.greenend.org.uk/%7Esgtatham/putty>

9. **WebCleaner** [Linux] – Controla el tipo de datos que el servidor proxy analiza; útil para la eliminación de malware y anuncios popups.

Site: <http://webcleaner.sourceforge.net>

10. **Cyberduck** [Mac] – Usa Ssh para transferir archivos en forma segura y permite integrarse con características del OS X como Bonjour, Keychain, iDisk y mas.

Site: <http://cyberduck.ch>

12.7. Redes

1. **SoftPerfect Network Scanner** [Windows] – Muestra información acerca de la configuración de la red.

Site: <http://www.softperfect.com/products/networkscanner>

2. **Colasoft Packet Player** [Windows] – Captura paquetes y rastrea su traza a través de la red. Muy útil para encontrar los cuellos de botella y puntos vulnerables. Puede analizar trazas de paquetes realizadas por sniffers como Colasoft Capsa, Ethereal, Network General Sniffer y WildPackets EtherPeek/OmniPeek.

Site: http://www.colasoft.com/packet_player/index.php

3. **DomainScan Light** [Windows] – Monitorea puertos abiertos, unidades conectadas y cambios hechos en la red.

Site: <http://ghsoftware.dk/dsl/download.asp>

4. **SmoothWall** [Linux] – Sistema de Firewall, IDS y VPN.

Site: <http://smoothwall.org>

5. **Nagios** [Linux] – Sistema basado en web que permite el monitoreo de redes, servicios y hosts, configurable mediante un sistema de pluggins escritos en Perl o C. Permite dar aviso al administrador ante una advertencia.
Site: <http://www.nagios.org>
6. **nLive Core** [Linux] – Monitor de tráfico de red que funciona a través del análisis, inspección y filtrado de paquetes.
Site: http://www.vigiliti.com/index.php?option=com_content&task=view&id=67&Itemid=137
7. **Network Mapper** [Windows, Linux, Mac] – Herramienta para la exploración de red y auditoria de seguridad. Analiza paquetes para averiguar qué computadoras están disponibles, qué servicios ofrece y qué tipo de filtro de paquetes o firewall se están utilizando en los mismos.
Site: <http://insecure.org/nmap>
8. **Wireshark** [Windows, Linux, Mac] – Herramienta de captura de tráfico de red de tipo sniffer. Contiene más de veinticinco métodos para análisis de paquetes. Soporta captura en modo promiscuo.
Site: <http://www.wireshark.org>
9. **Nessus** [Windows, Linux, Mac] – Escaner de vulnerabilidades altamente escalable y muy completo gracias a su implementación mediante el uso de pluggins.
Site: <http://www.nessus.org/nessus>
10. **dSniff Control** [Mac] – Herramienta que pone a pruebas la seguridad de la red. Prueba ataques de sniffing y spoofing para obtener información.
Site: <http://www.mrwsoftware.com/software>
11. **Flame** [Mac] – Herramienta para análisis de estatus y eventos que ocurren en la red.
Site: <http://husk.org/apps/flame>

12. **Server Admin Tools** [Mac] – Es la solución de Apple para garantizar y supervisar una red.

Site: <http://www.apple.com/support/downloads/serveradmintools1047.html>

12.8. Sistemas de detección de intrusos

1. **HenWen** [Mac] – Efectiva implementación del IDS Snort para usuarios de Mac.

Site: <http://seiryu.home.comcast.net/~seiryu/henwen.html>

2. **Open Source Host-based Intrusion Detection System** [Windows, Linux, Mac]
– IDS personal que analiza logs, monitorea el registro de Windows, realiza chequeos de integridad, detecta malware y rootkits, y puede generar alertas y respuestas en tiempo real.

Site: <http://www.ossec.net>

3. **Prelude** [Linux] – Sistema de detección de anomalías que genera reportes de usuario.

Site: <http://www.prelude-ids.org>

4. **Snort** [Windows, Linux, Mac] – La primer solución de IDS libre, detecta una amplia variedad de ataques, incluyendo desbordamientos de memoria, fingerprinting, escaneo de CGI scans y más. Escanea y analiza el tráfico en tiempo real en una red IP.

Site: <http://www.snort.org>

12.9. VPN (Virtual Private Network)

1. **SSL-Explorer** [Windows, Linux, Mac] – VPN altamente escalable, que se integra directamente con el browser y solo requiere un puerto abierto para trabajar.

Site: <http://3sp.com/showSslExplorerCommunity.do?referrer=sslexplorer>

2. **OpenVPN** [Windows, Linux, Mac] – Solución de VPN enfocada en el uso de acceso remoto. Recomendada para securizar redes WiFi.

Site: <http://openvpn.net>

3. **IPSecuritas** [Mac] – VPN basada en IPsec que permite la transferencia segura de datos.

Site: <http://www.lobotomo.com/products/IPSecuritas>

4. **strongSwan** [Linux] – VPN basado en IPsec que puede utilizar diferentes métodos de encriptación.

Site: <http://www.strongswan.org>

12.10. Archivos Temporales

1. **HijackThis** [Windows] – Escanea los exploradores, archivo de registro y configuración del sistema operativo para generar un reporte del estado de la computadora.

Site: http://www.trendsecure.com/portal/en-US/tools/security_tools/hijackthis

2. **Zappit System Cleaner** [Windows] – Elimina las huellas luego de usar la computadora, incluyendo: archivos temporales, documentos recientes, búsquedas realizadas, comandos ejecutados, temporales y formularios de Internet, histórico y mucho más.

Site: <http://zappit.net>

3. **CCleaner** [Windows] – Un sistema de optimización y a su vez una herramienta de privacidad que limpia el archivo de registro, archivos temporarios, cookies, historial de Internet, cache, y más.

Site: <http://www.ccleaner.com>

4. **Advanced WindowsCare v2 Personal** [Windows] – Limpia archivos temporarios y el archivo de registro de Windows para garantizar la máxima privacidad. También actúa como removedor de spyware.

Site: <http://www.iobit.com/advancedwindowscareper.html>

5. **EasyCleaner** [Windows] – Herramienta que realiza limpieza del archivo de registro de Windows registry y huellas que puedan haber quedado en la computadora.

Site: <http://personal.inet.fi/business/toniarts/ecleane.htm>

6. **CleanCache** [Windows] – Elimina archivos temporales de Internet y todo lo que pueda dejar huellas de por donde se estuvo navegando.

Site: <http://www.buttuglysoftware.com/CleanCache3.html>

7. **xpy** [Windows] – Herramienta que reconfigura la instalación por default de Windows para mejorar algunas características del sistema operativo.

Site: <http://nsis.whye.org/xpy>

8. **Cookie Eraser** [Windows] – Herramienta que ayuda a identificar y eliminar cookies innecesarias.

Site: <http://www.cookie-eraser.com>

9. **Yet Another System Utility** [Mac] – Limpieza de la cache, el historial de Internet, y otras áreas donde pueda haber huellas de uso de la computadora.

Site: <http://jimmitchell.org/projects/yasu>

12.11. Wireless

1. **Kismet** [Linux, Mac] – Uno de los más populares detectores de redes wireless, sniffer e IDS.

Site: <http://www.kismetwireless.net>

2. **AirSnare** [Windows] – IDS wireless que alerta cuando una MAC Address hostil está intentando acceder a su red

Site: <http://home.comcast.net/%7Ejay.deboer/airsnare>

3. **myWIFIzone** [Windows] – Actúa como gateway que bloquea usuarios que intenten conectarse a su red WiFi sin que tengan permiso expreso suyo.

Site: <http://www.mywifizone.com>

4. **iOpus Private Internet Gateway** [Windows] – Crea un túnel seguro que encripta las comunicaciones de su red WiFi.

Site: <http://www.iopus.com/iPig>

5. **LucidLink WiFi Client** [Windows] – Simplifica el método de conexión a una red WiFi y advierte si está intentando unirse a una red insegura.
Site: http://www.lucidlink.com/lucidlink_wifi_client.htm
6. **AirTrafficControl** [Mac] – Monitorea las conexiones WiFi verificando calidad y seguridad de las mismas.
Site: <http://www.spintriplet.com/atc>

12.12. Encriptación

1. **Pastor** [Mac] – Simple herramienta que sirve para almacenar todos los usuarios y contraseñas del sistema.
Site: <http://www.mehlau.net/pastor>
2. **Access Manager** [Windows] – Almacena todas las contraseñas y la información sensible en un archivo seguro.
Site: <http://www.accessmanager.co.uk>
3. **RememberMe** [Windows] – Almacena sus usuarios y contraseñas en un archivo encriptado con AES de 128 bits.
Site: <http://capp-ware.com/rememberme.htm>
4. **Kruptos 2** [Windows] – Encriptador de archivos de fácil uso integrado con el explorador de Windows mediante el uso del menú contextual del botón derecho del mouse.
Site: <http://www.kruptos2.co.uk>
5. **Crypt4Free** [Windows] – Utiliza Blowfish y DESX para proteger archivos y encriptar mensajes de chat y mails.
Site: http://www.secureaction.com/encryption_free
6. **PicSecret** [Web, Mac] – Permite enviar mensajes cifrados disfrazados de imágenes ordinarias.
Site: <http://www.picsecret.com/index.html>

7. **GNU Privacy Guard** [Windows, Linux, Mac] – Herramienta de encriptación de línea de comandos que utiliza varios tipos de cifrado como OpenPGP, AES, SHA-1, y mas.

Site: <http://www.gnupg.org>

8. **TrueCrypt** [Windows, Linux] – Crea y maneja unidades virtuales encriptadas.

Site: <http://www.truecrypt.org>

12.13. Otros

1. **Active Security Monitor** [Windows] – Utiliza un sistema de pruebas para determinar si se encuentra en un ambiente seguro.

Site: <http://free.aol.com/tryaolfree/index3.adp?promo=808524>

2. **PC Security Test** [Windows] – Determina las vulnerabilidades de su sistema sometiéndolo a diferentes tipos de ataque simulados.

Site: <http://www.pc-st.com/us/index.htm>

3. **PCPal** [Windows] – Detecta vulnerabilidades de seguridad en su sistema y le aconseja sobre cómo reducir sus riesgos.

Site: <http://www.pcpal.com/index.htm>

4. **Simple File Shredder** [Windows] – Elimina de forma segura y permanente archivos de su disco.

Site: <http://www.scar5.com>

5. **BCWipe** [Windows] – Al igual que el anterior, elimina de forma permanente archivos de su disco.

Site: <http://www.jetico.com/index.htm#/bcwipe3.htm>

6. **Permanent Eraser** [Mac] – Al igual que la herramienta anterior, elimina archivos de forma permanente.

Site: <http://www.edenwaith.com/products/permanent%20eraser>

7. **S3 Change Explorer** [Windows] – Gráficamente sigue la pista de los cambios

realizados en su sistema; útil para detectar si un usuario o archivo está haciendo cambios indeseados en ficheros sensibles.

Site: http://solidcore.com/learn/s3_change_explorer.html

8. **Process Monitor** [Windows] – Solución de Microsoft para monitorear todos los procesos que corren en su computadora.

Site: <http://www.microsoft.com/technet/sysinternals/utilities/processmonitor.msp>

9. **Check Failed Password Attempts** [Mac] – Herramienta que sirve para analizar los intentos de ingresos fallidos con la cuenta administrador.

Site: <http://www.jonn8.com/html/FailedPWAttempts.html>

10. **CheckMate** [Mac] – Compara checksums de algunos archivos para determinar su integridad antes de ser descargado.

Site: http://personalpages.tds.net/%7Ebrian_hill/checkmate.html

11. **Checksum Validator** [Mac] – Una muy simple herramienta de verificación y comparación de checksums .

Site: <http://www.kingdom-era.com/products/validator>

12. **SysInternals Suit** [Windows] – Suit de pequeñas herramientas que permiten administrar y analizar distintas áreas de la computadora (como *Filemon*, o monitor de lectura/escritura de archivos, *Regmon* o monitor de registro de Windows,). Se recomienda descargas desde este link o mediante P2P ya que a mitad del 2007 Microsoft compró dicha suit y es probable que varias funcionalidades hayan sido eliminadas.

Site: <http://sysinternals-suite.uptodown.com>

13. Anexo VI – Sitios de Noticias de Seguridad

1. Tecnoticias

Site: <http://www.tecnoticias.info/>

Rss: http://tecnoticias.info/index2.php?option=com_rss&feed=RSS2.0&no_html=1

2. kriptopolis

Site: <http://www.kriptopolis.org/>

Rss: <http://www.kriptopolis.org/atom/feed>

3. Hispasec

Site: <http://www.hispasec.com/>

Rss: <http://www.hispasec.com/rss/unaaldia.xml>

4. NoticiasTech

Site: <http://noticiastech.com>

Rss: <http://noticiastech.com/wordpress/?feed=rss2>

5. OSNews

Site: <http://www.osnews.com/>

Rss: <http://osnews.com/files/recent.xml>

6. News.com

Site: <http://news.com.com>

Rss: http://www.news.com/2547-1_3-0-20.xml

7. ComputerWorld

Site: <http://www.computerworld.com/news/>

Rss: <http://feeds.computerworld.com/Computerworld/TopNews>

8. FayerWayer

Site: <http://www.fayerwayer.com/>

Rss: <http://feeds.feedburner.com/fayerwayer>

9. HispaLinux

Site: <http://www.hispalinux.es>

Rss: <http://www.hispalinux.es/node/feed>

10. SecurityFocus

Site: <http://www.securityfocus.com/>

Rss: <http://www.securityfocus.com/rss/news.xml>

11. diarioTI

Site: <http://www.diarioti.com/>

Rss: http://www.diarioti.com/inc/titulares_rss.xml

12. Barrapunto

Site: <http://barrapunto.com/>

Rss: <http://barrapunto.com/barrapunto.rss>

13. Baquia

Site: <http://www.baquia.com/>

Rss: <http://www.baquia.com/feeds/feed.php?formato=rss20&modo=noticias>

14. VivaLinux

Site: <http://www.vivalinux.com.ar/>

Rss: <http://www.vivalinux.com.ar/xml/rss>

15. TodoLinux

Site: <http://www.todo-linux.com/>

Rss: <http://www.todo-linux.com/backend.php>

16. Cafelug

Site: <http://www.cafelug.org.ar/modules/news/>

Rss: <http://www.cafelug.org.ar/backend.php>

17. Infoworld

Site: <http://www.infoworld.com/news/>

Rss: <http://www.infoworld.com/rss/news.xml>

18. Softpedia

<http://news.softpedia.com/>

19. Eweek

Site: <http://www.eweek.com/>

Rss: <http://feeds.ziffdavisenterprise.com/RSS/tech>

20. InformationWeek

Site: <http://www.informationweek.com/>

Rss: <http://www.informationweek.com/rss/all.xml>

21. LinuxInsider

Site: <http://linuxinsider.com/>

Rss: <http://linuxinsider.com/perl/syndication/rssfull.pl>

22. NewsFactor

Site: <http://www.newsfactor.com/>

Rss: <http://www.newsfactor.com/xml/rss/syndication.xml>

23. Extremetech

Site: <http://www.extremetech.com/>

Rss: <http://feeds.ziffdavis.com/ziffdavis/extremetech>

24. Arstechnica

Site: <http://arstechnica.com/>

Rss: <http://feeds.arstechnica.com/arstechnica/BAaf>

25. Noticias.com

Site: <http://www.noticias.com/>

Rss: <http://www.noticias.com/rss/tecnologia.rss>

26. The Inquirer Español

Site: <http://es.theinquirer.net/>

Rss: <http://www.theinquirer.es/feed/atom>

27. LaFlecha

Site: <http://www.laflecha.net/>

Rss: <http://www.laflecha.net/laflecha.rss>

28. LinuxWatch

Site: <http://www.linux-watch.com/>

Rss: <http://www.linux-watch.com/backend/headlines.rss>

29. NetworkWorld

Site: <http://www.networkworld.com/>

Rss: <http://www.networkworld.com/rss/convergence.xml>

30. WiMaxNetNews

Site: <http://wimaxnetnews.com/>

Rss: <http://wimaxnetnews.com/rss2.xml>

31. Infosecurity

Site: <http://www.infosecurity-magazine.com/news/>

Rss: <http://www.infosecurity-magazine.com/RSS/LiveFeed.xml>

32. SecuriTeam

Site: <http://www.securiteam.com>

Rss: <http://www.securiteam.com/securiteam.rss>

33. FWBuilder

Site: <http://www.fwbuilder.org/>

Rss: <http://www.fwbuilder.org/index.rdf>

34. Windows Security

Site: <http://www.windowsecurity.com/>

Rss: <http://rss.windowsecurity.com/allnews.xml>

35. Astalavista News

Site: <http://www.astalavista.com/index.php?section=newsletter>

Rss: http://www.astalavista.com/feed/news_headlines_en.xml

36. IpCop

Site: <http://www.ipcop.org/>

Rss: <http://www.ipcop.org/backend.php>

37. Sahana.lk

Site: <http://www.sahana.lk/>

Rss: <http://www.sahana.lk/node/feed>

38. Argeniss

Site: <http://www.argeniss.com/>

39. Identidad Robada

Site: <http://identidadrobada.com/>

Rss: <http://www.identidadrobada.com/site/rss/feed.php?canal=novedades>

40. InformationShield

Site: <http://www.informationshield.com/>

MailList: <http://www.informationshield.com/newsletter.html>

41. RootSecure

Site: <http://www.rootsecure.net/>

Rss: http://www.rootsecure.net/?p=rss_xml_feed_ft&i=20

Bibliografía

- [ACKPRIOR] - <http://www.benzedrine.cx/ackpri.html>
- [ALBA] - <http://www.linuxalbacete.org/articulos/hs/hs6.htm>
- [BSDSCRU] - <http://www.openbsd.org/faq/pf/es/scrub.html>
- [BUILDFW] - Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, Building Internet Firewalls, 2000
- [CAHO06] - Earl Carter, Jonathan Hogue, Intrusion Prevention Fundamentals , 2006
- [CALIJAPO97] - Tetsuichi Asaka, Manual de herramientas de calidad: El enfoque japonés, 1997
- [CANA01] - John E. Canavan, Fundamentals of Network Security, 2001
- [CARP] - <http://www.pastelero.net/2007/01/25/soekris-net4801-openbsd-40-carp>
- [CARROLLIA] - <http://www.mensa.es/carrollia/c79.pdf>
- [CARVANTES] - <http://www.caravantes.com/pgp/pgp.htm>
- [CHRE] - Chad Renfro, Basic Packet-Sniffer Construction from the Ground Up,
- [CISC07] - Cisco, Cisco IOS NetFlow, 2007
- [COD23] - http://dev.codigo23.net/documentacion/OpenBSD/openbsd_pf/openbsd_pf_html
- [COGE04] - Kerry J. Cox, Christopher Gerg, Managing Security with Snort and IDS Tools,
- [COL_Stevens] - <http://www.win.tue.nl/hashclash/fastcoll.pdf>
- [COLL_PRED] - <http://www.win.tue.nl/hashclash/Nostradamus/>
- [COLL-Alter] - <http://home.gwu.edu/~khenry/CSci381/AlternativeHashFunctions.pdf>
- [CollHash] - <http://eprint.iacr.org/2004/199.pdf>
- [COREWA1] - <http://www.geocities.com/SiliconValley/2072/aial.htm>
- [COREWA2] - http://www.anvari.org/fortune/Miscellaneous_Collections/54152_core-wars-n-a-game-between-assembler-programs-in-a-simulated-machine.html
- [CRIPTO] - Manuel J. Luce na Lópe z, Criptografía y Seguridad en Computadores,
- [CRIPTO.ES] - <http://www.cripto.es>
- [CRO_VIR] - Lic. Cristian Borghello, Cronología de los virus informáticos, 2006
- [CRYPTO_ELEC] - CriptoRed, Libro Electrónico de Seguridad Informática y Criptografía, 2006
- [CRYPTO_KHAN] - David Khan, The Codebreakers: The Story of Secret Writing, 1967
- [CRYPTOLABS] - <http://crypto.dsplabs.com.au/classical/railfence-cipher-cryptanalysis.php>
- [CSEM96] - Chuck Semeria, Internet Firewalls and Security., 1996
- [cybWar2007] - Juan Pablo Perez Etchegoyen, Wardriving Buenos Aires 2007, 2007
- [FINGER06] - http://64.233.169.104/search?q=cache:d00w1JBSV48J:blackspiral.org/docs/fingerprint.ps+instalar+Fingerprint+fucker&hl=es&ct=clnk&cd=5&gl=ar&lr=lang_es&client=firefox-a
- [FWOBSD] - http://wiki.sblug.org/Example_pf.conf_for_OpenBSD_Firewall
- [GNU4ALL] - <http://gnu4all.wordpress.com/2007/08/27/portsentry/>
- [GnuPG] - <http://www.gnupg.org/index.es.html>
- [GUSA_SRC] - <http://www.morrisworm.com/worm-src.tar.gz>

[GWOLF] - <http://gwolf.org/files/portsentry/node14.html>

[HACKPRO] - , Hack Proofing Linux: A Guide to Open Source Security, 2001

[HAMMU229] - <http://www.historiaclasica.com/2007/06/el-cdigo-de-hammurabi-leyes-201-250.html>

[HAMMURABI] - <http://www.historiaclasica.com/2007/05/el-cdigo-de-hammurabi.html>

[HIS_MOR] - <http://www.hispasec.com/unaaldia/7>

[HSRR00] - Hernandez, Sierra, Ribagorda, Ramos, Técnicas de detección de sniffers, 2000

[IDLESCAN] - <http://nmap.org/idlescan-es.html>

[KUSE99] - Kurt Seifried, Guía de Seguridad del Administrador de Linux, 1999

[MRAN93] - Marcus J. Ranum, Thinking about Firewalls., 1993

[NETSECMai] - Eric Maiwald, Network security: A Beginner's Guide, 2003

[NETSECREf] - Roberta Bragg, Keith Strassberg, MarkRhodes-Ousley, omplete Reference, 2004

[NMAP_IDLE] - <http://nmap.org/idlescan.html>

[OBSD_PF] - Varios, The Openbsd Pf Packet Filter Book, 2006

[OPENBSD.ORG] - <http://www.openbsd.org/>

[PASNIFF] - http://wapedia.mobi/es/Packet_sniffer

[PRIORACK] - http://www.openbsderos.org/wiki/index.php?title=Priorizando_ACK

[PRONAGIOS] - Pro Nagios 2.0, Pro Nagios 2.0, Pro Nagios 2.0

[RFC3164] - <http://www.ietf.org/rfc/rfc3164.txt>

[RYSP03] - Ryan Spangler, Packet Sniffer Detection with AntiSniff, 2003

[SAMPPF] - <http://www.openbsd.org/faq/pf/example1.html>

[SCHI97] - John Schimmel., A historical look at firewall technologies., 1997

[SCIRET] - http://www.e-compugraf.com/sciret/index.php?view=ViewArticle&id=7&set=all&sort=views_asc

[SECFOC] - <http://www.securityfocus.com/infocus/1580>

[SEGPROBSD] - http://dev.codigo23.net/documentacion/OpenBSD/openbsd_pf/openbsd_pf.html

[SIHA95] - Karanjit Siyan and Chris Hare, Internet y seguridad en redes., 1995

[syslogNg] - <http://www.balabit.com/network-security/syslog-ng/>

[VIHU02] - Antonio Villalon Huerta, SEGURIDAD EN UNIX YREDES, 2002

[WACA94] - John P. Wack and Lisa J. Carnahan, .Keeping your site comfortably secure: an introduction to Internet Firewalls., 1994

[WIKI_3DES] - <http://es.wikipedia.org/wiki/3DES>

[WIKI_AES] - http://es.wikipedia.org/wiki/Advanced_Encryption_Standard

[WIKI_ASIM] - http://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica

[WIKI_CRYPT] - <http://es.wikipedia.org/wiki/Criptograf%C3%ADa>

[WIKI_DMZ] - http://es.wikipedia.org/wiki/Zona_desmilitarizada

[WIKI_ENIG] - http://es.wikipedia.org/wiki/Enigma_%28m%C3%A1quina%29

[WIKI_ESTE] - <http://es.wikipedia.org/wiki/Esteganograf%C3%ADa>

[WIKI_FRIED] - http://en.wikipedia.org/wiki/William_F._Friedman

[WIKI_GPG] - <http://es.wikipedia.org/wiki/Gnupg>

[WIKI_HELLMAN] - http://en.wikipedia.org/wiki/Martin_Hellman
[WIKI_ITAR] - http://en.wikipedia.org/wiki/International_Traffic_in_Arms_Regulations
[WIKI_KASI] - http://es.wikipedia.org/wiki/M%C3%A9todo_Kasiski
[WIKI_MW] - http://en.wikipedia.org/wiki/Morris_Worm
[WIKI_nTop] - <http://es.wikipedia.org/wiki/Ntop>
[WIKI_PGP] - http://es.wikipedia.org/wiki/Pretty_Good_Privacy
[WIKI_PHIL] - http://es.wikipedia.org/wiki/Phil_Zimmermann
[WIKI_RSA] - <http://es.wikipedia.org/wiki/RSA>
[WIKI_SHA] - <http://es.wikipedia.org/wiki/SHA>
[WIKI_SHANNON] - http://en.wikipedia.org/wiki/Claude_Shannon
[WIKI_TAYLOR] - http://es.wikipedia.org/wiki/Frederick_W._Taylor
[WIKI_TLS] - http://en.wikipedia.org/wiki/Secure_Sockets_Layer
[WIKI_VIGE] - http://es.wikipedia.org/wiki/Cifrado_de_Vigen%C3%A9
[WIKIOBSD] - <http://es.wikipedia.org/wiki/OpenBSD>
[WIKIOPENBSD] - <http://wiki.e-shell.org/OpenBSDSeguridadProactivaYFirewallsEs>
[WIKISNIF] - http://es.wikipedia.org/wiki/Tipos_de_Sniffer
[WWWDrv] - www.worldwidewardrive.org
[WWWNAGIOS2] - http://nagios.sourceforge.net/docs/2_0/toc.html
[WWWNAGIOSES] - <http://nagios.linuxbaja.org/>
[WWWNMAP] - <http://readhacked.wordpress.com/2007/08/07/nmap-guia-introduccion-facil/>
[WWWOBSD] - <http://www.openbsd.org>
[WWWSEGINF] - <http://www.segu-info.com.ar/firewall/firewall.htm>
[WWWSP] - <http://www.securitypark.co.uk/article.asp?articleid=26330&CategoryID=1>
[WWWTEXCI] - <http://www.textoscientificos.com/redes/firewalls-distribuidos/firewalls/convencionales>
[WWWWLEARN] - http://www.wikilearning.com/tutorial/seguridad_en_unix_y_redes-arquitecturas_de_cortafuegos/9777-87
[ZIMMERMANN] - <http://www.philzimmermann.com/ES/background/index.html>

14. Glosario

1. **Acknowledgement** (o acuse de recibo), es un mensaje que se envía para confirmar que otro mensaje o un conjunto de ellos han llegado.
2. **DMZ** (Demilitarized zone) o Zona Desmilitarizada es una red local que se ubica entre la red interna de una organización y una red externa, generalmente Internet.
3. **E-token** es un pequeño generador digital de códigos, que protege sus operaciones bancarias online. Estos códigos se modifican en forma constante y nunca se repiten, ofreciéndole mayor seguridad a la hora de manejar sus cuentas a través de Internet. Cada e-Token es único, personal e intransferible.
4. **Firewall** o Cortafuegos: Componente de hardware o software utilizado en una red para controlar las comunicaciones, permitiendo o prohibiendo el tráfico de red según las políticas del sistema.
5. **GPG** o GNU Privacy Guard: Herramienta para cifrado y firmas digitales
6. **Honeypot** o “tarro de miel”, es un software o conjunto de computadoras cuya intención es atraer y distraer a los atacantes simulando ser sistemas débiles para recolectar información de los mismos y así poder conocerlos y reaccionar con medidas defensivas.
7. **Keyloggers** o Registrador de teclas, es un malware del tipo daemon que se encarga de registrar las pulsaciones que se realizan sobre el teclado, para escribirlas en un archivo y generalmente enviarlas a través de Internet.
8. **LAN** (Local Area Network) o red de área local: Interconexión de varias computadoras y periféricos para que se comuniquen entre sí generalmente no más de 100 metros.
9. **Man-in-the-middle** o MitM, es un ataque en el que cual se adquiere la capacidad de leer, insertar y modificar a voluntad, los mensajes entre las partes.
10. **NAT** (Network Address Translation o Traducción de Dirección de Red) es un mecanismo utilizado por los routers para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles convirtiendo en tiempo real las direcciones utilizadas en los paquetes transportados.
11. **Netflow** es un protocolo de red desarrollado por Cisco para recolectar información del tráfico IP.
12. **OSI** (Open System Interconnection): Modelo en capas de interconexión de Sistemas Abiertos .
13. **P2P** (Peer to peer) o punto a punto: Es una red de pares que no posee una parte cliente y otra servidores sino que ambas partes se comportan de ambas maneras.
14. **PIX** o Private Internet Exchange: Solución de seguridad ofrecida por Cisco Systems el cual implementa un firewall por hardware.
15. **Proxy**: Aplicación que intercepta las conexiones de red entre los clientes y destinos, comúnmente utilizado para la interconexión de múltiples computadoras a un único vínculo de Internet aprovechando una cache de contenidos que es compartido entre todos los equipos. con la consiguiente mejora en los tiempos de acceso a consultas coincidentes y al mismo tiempo liberando la carga del enlace hacia Internet.
16. **Puerta Trasera** o BackDoor: Secuencia especial dentro del código de un programa colocado adrede mediante el cual, el desarrollador puede acceder o saltar partes de la secuencia normal del programa utilizado en períodos de prueba o ante una emergencia.
17. **Router** o enrutador, es un dispositivo para interconexión de redes de computadoras que opera en la capa tres (capa de red).
18. **Secure Sockets Layer** (SSL) y Transport Layer Security (TLS) son protocolos criptográficos que proporcionan comunicaciones seguras en Internet mediante el uso de criptografía [WIKI_TLS].
19. **Spoofing** hace referencia al uso de técnicas de suplantación de identidad dentro de una comunicación de red.
20. **Stealth** Técnica utilizada por algunos virus que esconde las modificaciones hechas a las claves de registro, archivos del sistema o cualquier propiedad que indique que está infectado. Esta técnica hace uso de todos los medios posibles para que la presencia del virus pase totalmente inadvertida, anulando efectos tales como los atributos de los archivos (tamaño, fecha/hora de modificación, etc.), y hasta el uso de la memoria RAM.
21. **WAN** (Wide Area Network) o Red de área amplia: Tipo de red de computadoras donde se cubren distancias de hasta varios kilómetros para interconexión de redes o equipos terminales distanciados en un área geográficamente amplia.
22. **Worm** o Gusano: virus informático que tiene la propiedad de duplicarse a sí mismo infectando toda la red de computadoras. A diferencia de los virus, éstos no precisan alterar los archivos de programas, ya que pueden residir en memoria, enviar copias de sí mismo a otros nodos y así consumir los recursos del sistema.